EUROPEAN SOUTHERN OBSERVATORY

Organisation Européenne pour des Recherches Astronomiques dans l'Hémisphère Austral

Europäische Organisation für astronomische Forschung in der südlichen Hemisphäre

VERY LARGE TELESCOPE

CRIRES Pipeline User Manual

VLT-MAN-ESO-19500-4406

Issue 1.14

Date May 20th 2021

Prepared:	L. Luvaul	May 20th 2021	
	Name	Date	Signature
Approved:	P. Ballester		
	Name	Date	Signature
Released:	M. Peron		
	Name	Date	Signature

No content on this page

Change record

Issue/Rev.	Date	Section/Parag. affected	Reason/Initiation/Documents/Remarks
1.0	01/10/2007	All	For CRIRES 1.5.0
1.1	01/04/2008	All	For CRIRES 1.6.0
1.2	01/06/2008	All	For CRIRES 1.7.0
1.3	01/04/2009	All	For CRIRES 1.10.1
1.4	01/08/2009	All	For CRIRES 1.11.0
1.5	10/06/2010	All	For CRIRES 2.0.0
1.6	01/04/2011	All	For CRIRES 2.1.1
1.7	02/05/2011	All	For CRIRES 2.1.2
1.8	11/05/2011	All	For CRIRES 2.1.3
1.9	23/11/2011	All	For CRIRES 2.2.0
1.10	10/05/2012	All	For CRIRES 2.2.1
1.11	25/09/2012	All	For CRIRES 2.3.0
1.12	9/10/2012	All	For CRIRES 2.3.1
1.13	17/02/2014	All	For CRIRES 2.3.2
1.14	20/05/2021	All	For CRIRES 2.3.13

No content on this page

Contents

1 Introduction

1.1 Purpose

The CRIRES pipeline is a subsystem of the *VLT Data Flow System* (DFS). It is used in two operational environments, for the ESO *Data Flow Operations* (DFO), and for the *Paranal Science Operations* (PSO), in the quick-look assessment of data, in the generation of master calibration data, in the reduction of scientific exposures, and in the data quality control. Additionally, the CRIRES pipeline recipes are made public to the user community, to allow a more personalised processing of the data from the instrument. The purpose of this document is to describe a typical CRIRES data reduction sequence with the CRIRES pipeline.

This manual is a complete description of the data reduction recipes implemented by the CRIRES pipeline, reflecting the status of the CRIRES pipeline version 2.3.13.

1.2 Acknowledgements

Alain Smette and Jonathan Smoker, successively instrument scientists of CRIRES and Burkhard Wolff, QC group contact person have provided valuable feed-back on both the pipeline itself and its documentation, and made it possible to release it publicly. Paul Bristow and Florian Kerber have developed a powerful and precise model of the instrument that allows to calibrate the wavelength even for settings where calibration lines cannot be observed. On the side of the User Support Department, Michael Hilker is providing regular feedback which is both useful and appreciated.

1.3 Scope

Updated versions of the present document may be found on [6]. For general information about the current instrument pipelines status we remind the user of [3]. Quality control information are at [2].

Additional information on the Common Pipeline Library (CPL) and ESOREX can be found respectively at [5], [7]. The Gasgano tool is described in [10]. A description of the instrument is in [4]. The CRIRES instrument user manual is in [13].

1.4 Reference and applicable documents

[1] Konstantin Mirny Andrea Modigliani. *Detector monitoring recipes User Manual*. ESO/SDD/PSD, *ftp://ftp.eso.org/pub/dfs/pipelines/detmon/detmon-pipeline-manual-1.0.3.pdf*. VLT-MAN-ESO-19500-4846. 50, 78

[2] ESO/DMO/DFO, *http://www.eso.org/observing/dfo/quality/*. *Quality Control and Data Processing*. 11

[3] ESO/DMO/DFO, *http://www.eso.org/observing/dfo/quality/pipeline-status.html*. *VLT/VLTI Data Pipelines and Calibration Plans*. 11

[4] ESO/INS, *http://www.eso.org/sci/facilities/paranal/instruments/crires/*. *CRIRES instrument home page*. 11

[5] ESO/SDD/PSD, *http://www.eso.org/cpl/*. *CPL home page*. 11

[6] ESO/SDD/PSD, *http://www.eso.org/pipelines*. *CRIRES Pipeline Web Page*. 11

[7] ESO/SDD/PSD, *http://www.eso.org/cpl/esorex.html*. *ESOREX home page*. 11, 19, 20

[8] Kerber Florian et al. *Calibration of the ZnSe pre-disperser on ESO's cryogenic IR echelle spectrograph (CRIRES): comparison of the first results from CRIRES and the laboratory data from CHARMS.* ESO/SDD/PSD, proceedings of spie, 6269:626942-626942-12. spie, july. edition, 2006. 65

[9] Press William H. *Numerical recipes in C. The art of scientific computing*. ESO/SDD/PSD, cambridge: university press edition, 1989. 68

[10] D. Sforna M. Klein Gebbinck. *Gasgano User's Manual*. ESO/SDD/DFI ESO/SDD/SED, *http://www.eso.org/gasgano/*. VLT-PRO-ESO-19000-1932. 11, 13, 18, 25

[11] Ballester P. and M. R. Rosa. *Modeling echelle spectrographs*. ESO, astronomy and astrophysics supplement series 126, no. 3 (december): 563-571 edition, 1997. 64, 65

[12] Bristow Paul. *Model based wavelength calibration for CRIRES.* ESO/INS, proceedings of spie. vol. 627058. spie, july. edition. 64

[13] J.Smoker R. Siebenmorgen, A. Smette. *CRIRES User Manual*. ESO/INS ESO/PSO, *http://www.eso.org/instruments/crires/doc*. 11

2 Overview

In collaboration with instrument consortia, the Pipeline Department (PSD) of the Software Development Division (SDD) is implementing data reduction pipelines that have the following three main purposes:

Data quality control: pipelines are used to produce the quantitative information necessary to monitor instrument performance.

Master calibration product creation: pipelines are used to produce master calibration products (*e.g.*, combined bias frames, super-flats, wavelength dispersion solutions).

Science product creation: using pipeline-generated master calibration products, science products are produced for the supported instrument modes (*e.g.*, combined ISAAC jitter stacks; bias-corrected, flat-fielded FORS images, wavelength-calibrated UVES spectra). The accuracy of the science products is limited by the quality of the available master calibration products and by the algorithmic implementation of the pipelines themselves. In particular, adopted automatic reduction strategies may not be suitable or optimal for all scientific goals.

Instrument pipelines consist of a set of data processing modules that can be called from the command line, from the automatic data management tools available on Paranal or from Gasgano.

ESO offers two front-end applications for launching pipeline recipes, *Gasgano* [10] and *EsoRex*, both included in the pipeline distribution (see Appendix A). These applications can also be downloaded separately from `http://www.eso.org/gasgano` and `http://www.eso.org/cpl/esorex.html`. An illustrated introduction to Gasgano is provided in the "Quick Start" Section of this manual (see Section 4).

The CRIRES instrument and the different types of CRIRES raw frames and auxiliary data are described in Sections 3, 6, and 7.

A brief introduction to the usage of the available reduction recipes using Gasgano or EsoRex is presented in Section 4.

In section 5 we advice the user about known data reduction problems.

An overview of the data reduction, what are the input data, and the recipes involved in the calibration cascade is provided in Section 8.

More details on what are inputs, products, quality control measured quantities, and controlling parameters of each recipe is given in Section 9.

More detailed descriptions of the data reduction algorithms used by the individual pipeline recipes can be found in Section 10.

In Appendix A the installation of the CRIRES pipeline recipes is described and in Appendix C a list of used abbreviations and acronyms is given.

3 CRIRES Instrument Description

The VLT cryogenic high-resolution infrared echelle spectrograph CRIRES is located at the Nasmyth focus A of UT1 (Antu). It provides a resolving power of up to 100000 in the spectral range from 1 to 5 microns when used with a 0.2 arcsec slit. CRIRES can boost all scientific applications aiming at fainter objects, higher spatial (extended sources), spectral and temporal resolution. Simultaneous spectral coverage is maximized through a mosaic of four Aladdin III InSb arrays providing an effective 4096 x 512 focal plane detector array in the focal plane. Adaptive Optics (MACAO - Multi-Applications Curvature Adaptive optics) is used to optimize the signal-to-noise ratio and the spatial resolution.

It can be divided into four units:

- The fore-optics part provides the field rotation, cold pupil and field stops, adaptive optics and slit viewing.
- The prism pre-disperser isolates one echelle order and minimizes the total amount of light entering into the high-resolution section.
- The high-resolution section comprises the collimator, the echelle which is tilt-tuned for wavelength selection, the camera providing the 0."086/pixel scale, and the detectors.
- The calibration unit outside the cryogenic environment contains light sources for flux/wavelength calibration and detector flat-fielding.

It is offered for the first time in P79.

Figure 3.0.1 shows a picture of the instrument mounted on the first unit telescope on the VLT.

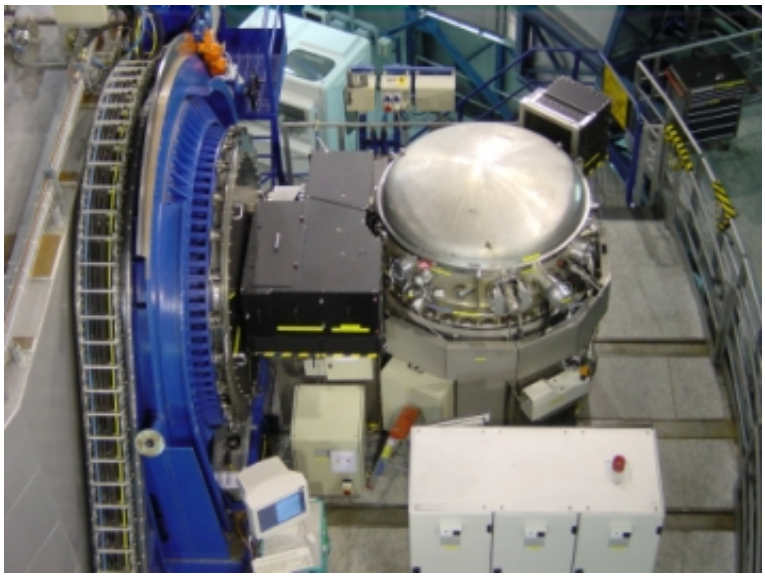

Figure 3.0.1: *Picture of CRIRES.*

Figure 3.0.2 shows the optical path of CRIRES.

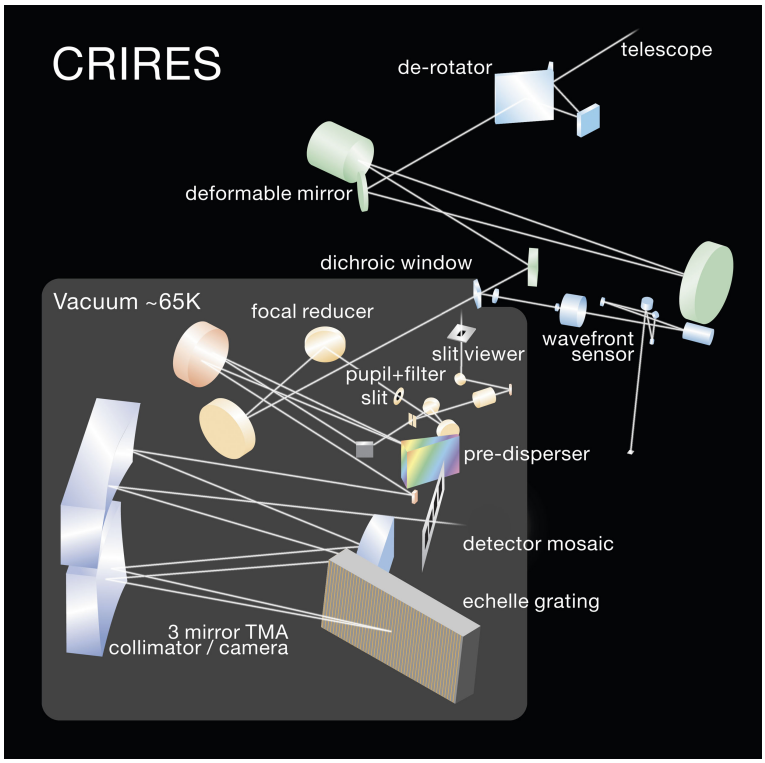

Figure 3.0.2: *Optical path of CRIRES.*

4 Quick start

This section describes the most immediate usage of the CRIRES pipeline recipes.

4.1 CRIRES pipeline recipes

The current CRIRES pipeline is based on a set of 12 stand-alone recipes involved in the data reduction cascade and 13 utilities.

The recipes are:

crires_spec_dark: Dark recipe

crires_spec_flat: Flatfield recipe

crires_spec_wavecal: Wavelength calibration

crires_spec_astro: Astrometry recipe

crires_spec_jitter: Direct observation recipe with or without jittering

crires_win_jitter: Observation recipe in Windowing mode

crires_win_dark: Dark recipe in Windowing Mode

crires_win_flat: Flatfield recipe in Windowing mode

crires_model_refine: Refining of the model configuration

crires_model_wlmap: Model Wavelength Map generation

crires_model_fix: Model improvement from user fixed wavelengths

detmon_ir_lg: DETMON Linearity/Gain recipe for the IR domain

The utilities are:

crires_util_extract: Spectrum extraction routine

crires_util_profile: Compute the FWHM of the spectrum along lambda

crires_util_combine: Images Combination tool

crires_util_wlcalib: Wavelength calibration of a spectrum

crires_util_wlinit: Wavelength calibration from the initial guess

crires_util_wlassign: Put the wavelength in the extracted table

crires_util_conversion: Compute the conversion factor

crires_util_sensitivity: Compute the sensitivity

crires_util_genlines: Generate spectrum calibration FITS tables

crires_util_genstd: Generate standard star FITS tables

crires_util_genconfig: Generate the Model Configuration tables

crires_util_genypos: Generate the Y Positions FITS table

crires_util_plot: Plotting tool for the CRIRES tables

4.2 An introduction to Gasgano and EsoRex

Before being able to call pipeline recipes on a set of data, the data must be opportunely classified, and associated with the appropriate calibrations. The *Data Classification* consists of tasks such as: "What kind of data am I?", *e.g.*, BIAS, "to which group do I belong?", *e.g.*, to a particular Observation Block or template. *Data Association* is the process of selecting appropriate calibration data for the reduction of a set of raw science frames. Typically, a set of frames can be associated if they share a number of properties, such as instrument and detector configuration. As all the required information is stored in the FITS headers, data association is based on a set of keywords (called "association keywords") and is specific to each type of calibration.

The process of data classification and association is known as data organisation. The *DO Category* is the label assigned to a data type as a result of data classification.

An instrument pipeline consists of a set of data processing modules that can be called from different host applications, either from the command line with *Esorex*, from the automatic data management tools available at the observatory, or from the graphical *Gasgano* tool.

Gasgano is a data management tool that simplifies the data organisation process, offering automatic data classification and making the data association easier (*even if automatic association of frames is not yet provided*). *Gasgano* determines the classification of a file by applying an instrument specific rule, while users must provide this information to the recipes when they are executed manually using *Esorex* from the command line. In addition, *Gasgano* allows the user to execute directly the pipeline recipes on a set of selected files.

4.2.1 Using Gasgano

To get familiar with the CRIRES pipeline recipes and their usage, it is advisable to begin with *Gasgano*, because it provides a complete graphic interface for data browsing, classification and association, and offers several other utilities such as easy access to recipes documentation and preferred data display tools.

Gasgano can be started from the system prompt in the following way:

```
gasgano &
```

The *Gasgano* main window will appear. On Figure 4.2.1, a view on a set of CRIRES data is shown as an example. *Gasgano* can be pointed to the directories where the data to be handled are located using the navigation panels accessible via the *Add/Remove Files* entry of the *File* menu (shown on the upper left of the figure).

More information about a single frame can be obtained by clicking on its name: the corresponding FITS file header will be displayed on the bottom panel, where specific keywords can be opportunely filtered and searched. Images and tables may be easily displayed using the viewers specified in the appropriate *Preferences* fields.

Frames can be selected from the main window for being processed by the appropriate recipe. This will open a *Gasgano* recipe execution window (see Figure 4.2.2), having all the specified files listed in its *Input Frames* panel.

Help about the recipe may be obtained from the *Help* menu. Before launching the recipe, its configuration may be opportunely modified on the *Parameters* panel (on top). The window contents might be saved for later use by selecting the *Save Current Settings* entry from the *File* menu, as shown in figure.

At this point the recipe can be launched by pressing the *Execute* button. Messages from the running recipe will appear on the *Log Messages* panel at bottom, and in case of successful completion the products will be listed on the *Output Frames* panel, where they can be easily viewed and located back on the Gasgano main window.

Please refer to the *Gasgano User's Manual* [10] for a more complete description of the *Gasgano* interface.

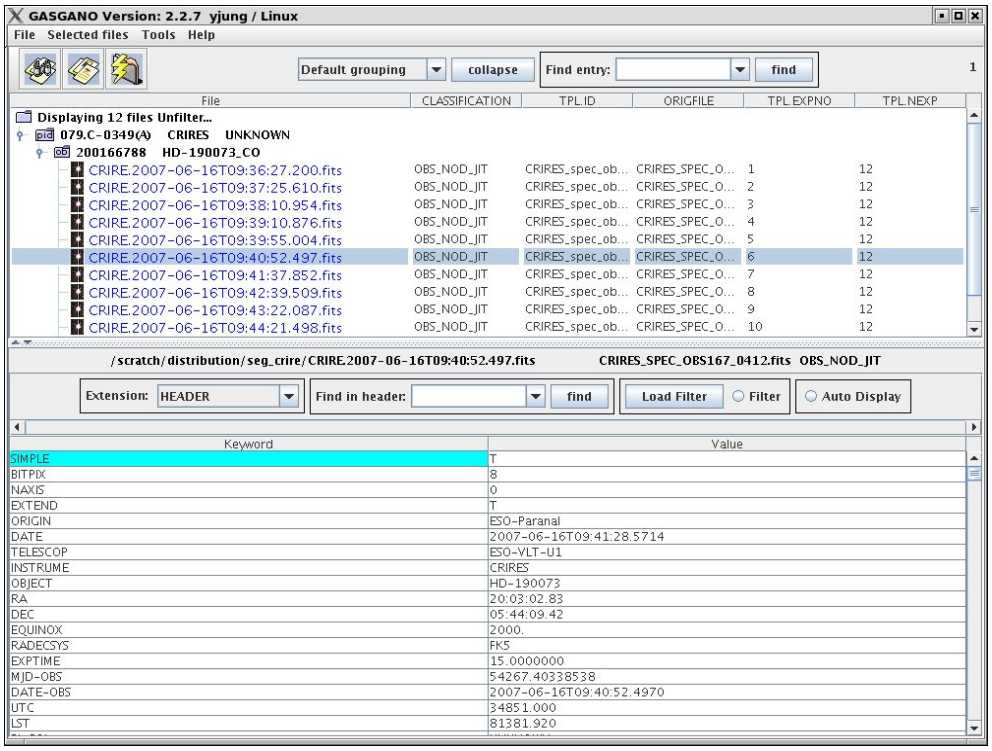

Figure 4.2.1: *The Gasgano main window.*

4.2.2 Using EsoRex

EsoRex is a command line utility for running pipeline recipes. It may be embedded by users into data reduction scripts for the automation of processing tasks. On the other side, *EsoRex* doesn't offer all the facilities available

with *Gasgano*, and the user must classify and associate the data using the information contained in the FITS header keywords (see Section 6). The user should also take care of defining the input set-of-frames and the appropriate configuration parameters for each recipe run:

The set-of-frames: Each pipeline recipe is run on a set of input FITS data files. When using *EsoRex* the file names must be listed together with their DO category in an ASCII file, the *set-of-frames* (SOF), that is required when launching a recipe. [1]

Here is an example of SOF, valid for the *crires_spec_jitter* recipe:

```
/file_path/CRIRE.2007-08-14T10:20:56.497.fits   OBS_DIR_JIT
/file_path/CRIRE.2007-08-14T10:22:44.285.fits   OBS_DIR_JIT
/file_path/flat.fits                            CALPRO_FLAT
/file_path/bpm.fits                             CALPRO_BPM
/file_path/lines_oh.fits                        CALPRO_OH_CATALOG
```

It contains for each input frame the full path file name and its DO category. The pipeline recipe will access the listed files when required by the reduction algorithm.

Using *Gasgano* as an interface to the pipeline recipes will always ensure a correct classification of all the data frames, assigning the appropriate DO category to each one of them (see Section 4.2.1).

EsoRex syntax: The basic syntax to use ESOREX is the following:

esorex [esorex_options] recipe_name [recipe_options] set_of_frames

To get more information on how to customise ESOREX (see also [7]) run the command:

esorex - -help

To generate a configuration file esorex.rc in the directory $HOME/.esorex run the command:

esorex - -create-config

A list of all available recipes, each with a one-line description, can be obtained using the command:

esorex - -recipes

All recipe parameters (aliases) and their default values can be displayed by the command

esorex - -params recipe_name

To get a brief description of each parameter meaning execute the command:

esorex - -help recipe_name

To get more details about the given recipe give the command at the shell prompt:

esorex - -man-page recipe_name

Recipe configuration: Each pipeline recipe may be assigned an *EsoRex* configuration file, containing the default values of the parameters related to that recipe.[2] The configuration files are normally generated in the directory `$HOME/.esorex`, and have the same name as the recipe to which they are related, with the file

[1] The set-of-frames corresponds to the *Input Frames* panel of the *Gasgano* recipe execution window (see Figure 4.2.2, page 21).

[2] The *EsoRex* recipe configuration file corresponds to the *Parameters* panel of the *Gasgano* recipe execution window (see Figure 4.2.2).

name extension `.rc`. For instance, the recipe *crires_spec_jitter* has its *EsoRex* generated configuration file named `crires_spec_jitter.rc`, and is generated with the command:

esorex --create-config crires_spec_jitter

The definition of one parameter of a recipe may look like this:

```
# --rej
# Left and right rejections.
crires.crires_spec_jitter.rej=0,10
```

In this example, the parameter `crires.crires_spec_jitter.rej` is set to the value `0,10`. In the configuration file generated by *EsoRex*, one or more comment lines are added containing information about the possible values of the parameter, and an alias that could be used as a command line option.

The recipes provided by the CRIRES pipeline are designed to implement a cascade of macro data reduction steps, each controlled by its own parameters. For this reason and to prevent parameter name clashes we specify as parameter prefix not only the instrument name but also the name of the step they refer to. Shorter parameter aliases are made available for use on the command line.

The command

esorex --create-config recipe_name

generates a default configuration file **recipe_name.rc** in the directory **$HOME/.esorex**[3].

A recipe configuration file different from the default one can be specified on the command line:

esorex --recipe-config=my_alternative_recipe_config

Recipe parameters are provided in Section 9 and their role is described in Section 10.

More than one configuration file may be maintained for the same recipe but, in order to be used, a configuration file not located under `$HOME/.esorex`, or having a name different from the recipe name, should be explicitly specified when launching a recipe.

Recipe execution: A recipe can be run by specifying its name to *EsoRex*, together with the name of a set-of-frames. For instance, the following command line would be used to run the recipe *crires_spec_jitter* for processing the files specified in the set-of-frames `crires_spec_jitter.sof`:

esorex crires_spec_jitter crires_spec_jitter.sof

The recipe parameters can be modified either by editing directly the used configuration file, or by specifying new parameter values on the command line using the command line options defined for this purpose. Such command line options should be inserted after the recipe name and before the SOF name, and they will supersede the system defaults and/or the configuration file settings. For instance, to set the *crires_spec_jitter* recipe *rej* parameter to `10,10`, the following should be typed:

esorex crires_spec_jitter --rej="10,10" crires_spec_jitter.sof

For more information on *EsoRex*, see [7].

[3] If a number of recipe parameters are specified on the command line, the given values will be used in the created configuration file.

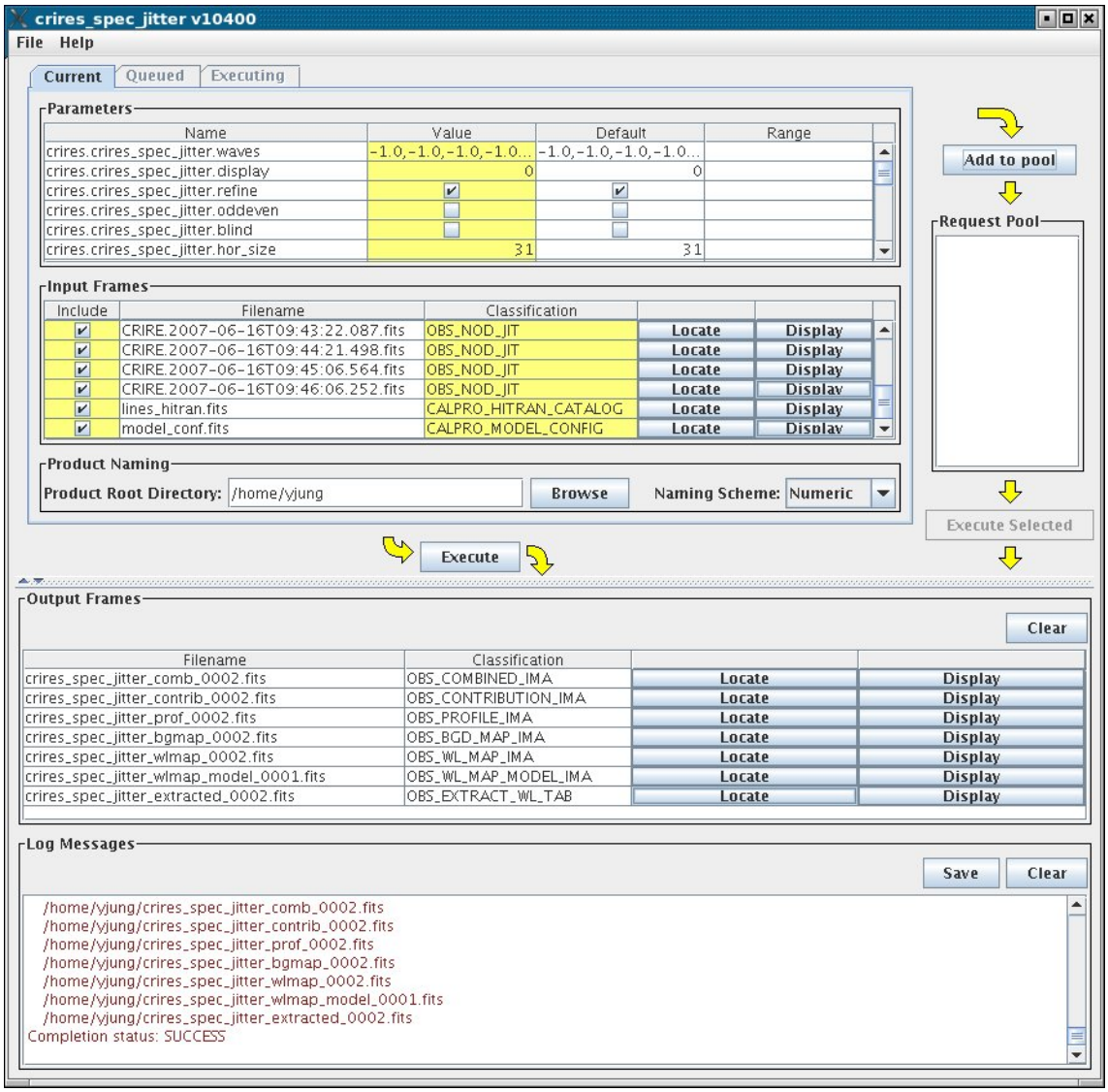

Figure 4.2.2: *The Gasgano recipe execution window.*

5 Known Problems

There are some features on the CRIRES data that are challenging for the data reduction pipeline, or simply that cannot be corrected by an automatic tool. For some of these, some additional interactive data analysis may be necessary to remove them.

Some of them are described in the following sections.

5.1 Lack of lines for the Wavelength calibration

Because of the high resolution of the instrument, accurate wavelength calibration is difficult in some wavelength settings. Indeed, in these cases, neither the sky spectrum nor the ThAr lamp spectrum offer the required 3 or 4 lines per detector necessary for a good wavelength calibration based on the cross-correlation technique described in Section 10.2.1. The calibration method based on the gas cell, on the ThAr lamp or the sky spectrum needs to be carefully chosen in each case.

In this version, the wavelength calibration based on the instrument model suffers from noisy prism and grating encoders. As a consequence, two successive observations with no change of wavelength setting may see significant differences in the wavelength zero point.

The figure 5.1.1 illustrated such problem.

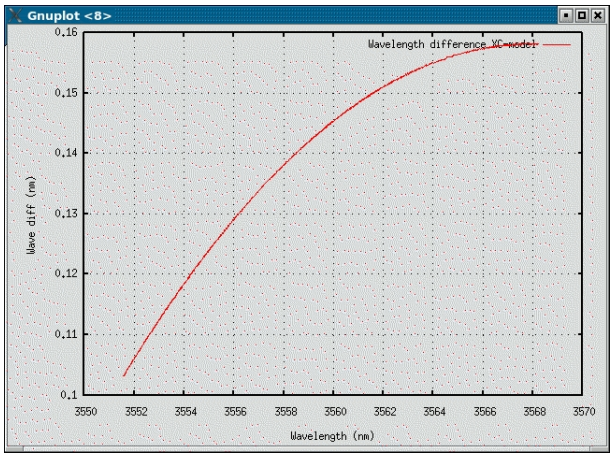

Figure 5.1.1: *Wavelength calibration difference with the model.*

This problem can be solved by anchoring the solution to at least a line (sky, ThAr, gas-cell) whose wavelength is precisely known. The utility *crires_model_fix* can then be used to correct the effect and obtain a better model configuration file.

5.2 Odd/even effect

The correction of detector non-linearity, including an "odd-even effect" (although different from the one seen in the detector of other instruments like ISAAC) relies on the use a flat-field itself corrected from non-linearity

effect. Such correction is very satisfactory providing both the flat-field and the science observations have been obtained with DIT of at least 2 seconds.

5.3 Distorsion

The pipeline does not correct for the effect of the distorsion introduced by the spectrograph. As a consequence, the extracted spectra obtained with e.g. the CRIRES_spec_obs_AutoNodOnSlit and the 0.2 arcsec slit will see their resolution slightly degraded as a result of the combination of the spectra obtained in the two nodding positions.

In order to keep the high resolution of the spectra, all image combination methods (in nodding) allow to generate the combined image using only *nodding A* position spectra or only *nodding B* position spectra, or both. Besides, the possibility is also offered to choose, amongst those 3 possible combined images, from which the spectrum is to be extracted. This way, the possibility is offered to extract the spectra from nodding data sets without degrading the spectrum resolution.

Since version 2.3.0, the wavalength calibration can be computed at several positions along the slit for each detector. The computed solutions are stored together with the position along the slit and the correlation factor in the wavelength calibration table (1 solution per row, 1 table/extension per detector). When the wavelength is applied to an extracted spectrum, the extracted spectrum position along the slit is used to derive the wavelength (by interpolation) from the closest wavelength solutions. Command line options are available to specify, for each detector, the list of positions along the slit where the wavelength needs to be calibrated.

5.4 Windowing

Only a part of the detectors was illuminated until May 2009. To avoid the contamination from the non illuminated parts, the pipeline loads systematically the illuminated parts, applies the data reduction tasks on those, and reconstructs the full detector (with 0s) when it comes to save the products. The illiminated area is known from the date the data were obtained.

The illuminated parts are changing with time, basically after each intervention on the instrument. The pipeline looks for the data date in its header and determines the appropriate illuminated zone to load.

So far the illuminated parts are defined like this for the period before february 5th 2009:

```
#define CRIRES_ILLUM_P1_LY1     75
#define CRIRES_ILLUM_P1_UY1     430
#define CRIRES_ILLUM_P1_LY2     75
#define CRIRES_ILLUM_P1_UY2     430
#define CRIRES_ILLUM_P1_LY3     65
#define CRIRES_ILLUM_P1_UY3     420
#define CRIRES_ILLUM_P1_LY4     55
#define CRIRES_ILLUM_P1_UY4     410
```

For the period between february 6th 2009 and may 28th 2009:

```
#define CRIRES_ILLUM_P2_LY1     11
#define CRIRES_ILLUM_P2_UY1     361
#define CRIRES_ILLUM_P2_LY2     12
#define CRIRES_ILLUM_P2_UY2     362
#define CRIRES_ILLUM_P2_LY3     1
#define CRIRES_ILLUM_P2_UY3     351
#define CRIRES_ILLUM_P2_LY4     1
#define CRIRES_ILLUM_P2_UY4     351
```

For the period after may 29th 2009, the whole detector is read.

6 CRIRES Data Description

This section gives a description of the raw data produced by CRIRES.

6.1 Data Format

The CRIRES instrument can produce data using two different read-out modes: *FowlerNsamp* and *FowlerNsampGRstWin*.

6.1.1 *FowlerNsamp* read-out mode

This type of data is the most commonly produced by the CRIRES instrument.

The CRIRES fits file contains a primary unit without data, and 4 extensions for the 4 chips. The EXTNAME keyword identifies which chip is stored in the current extension (CHIP3.INT1 for chip number 3). Each chip is a 1024x512 pixels image. The wavelength increases in each chip from left to right, and from chip 1 to chip 4. The gap between the chips corresponds to approximatively 280 pixels.

6.1.2 *FowlerNsampGRstWin* read-out mode

This is the so-called *windowing* mode or *stripes* mode. Like the previously described mode, there are 4 extensions for the 4 detectors. Detector 1 and 4 are ignored, the data are set to 0. Detector 2 and 3 both contain the readout of 2 horizontal stripes on the detector as described in Figure 6.1.1. The position and sizes of those stripes on the detector are determined by the some FITS keywords.

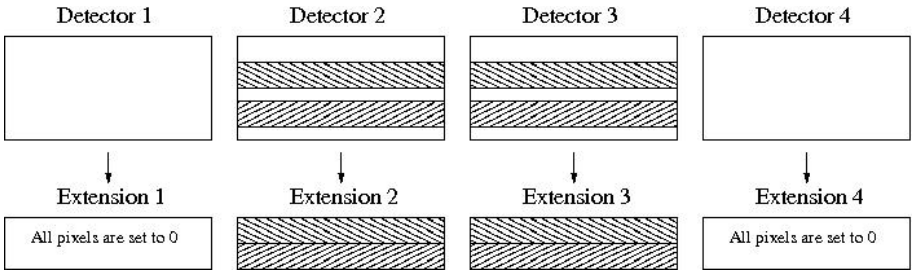

Figure 6.1.1: *FowlerNsampGRstWin read-out mode.*

6.2 Data classification

Any raw frame can be classified on the basis of a set of keywords read from its header. Data classification is typically carried out by the DO or by *Gasgano* [10], that apply the same set of classification rules. The association of a raw frame with calibration data (*e.g.*, of a science frame with a master bias frame) can be obtained by matching the values of a different set of header keywords.

Each kind of raw frame is typically associated to a single CRIRES pipeline recipe, *i.e.*, the recipe assigned to the reduction of that specific frame type. In the pipeline environment this recipe would be launched automatically.

In the following, all CRIRES raw data frames are listed, together with the keywords used for their classification and correct association. The indicated *DO category* is a label assigned to any data type after it has been classified, which is then used to identify the frames listed in the *Set of Frames* (see Section 4.2.2, page 19).

The classification described in the next section is defined in the *crire.oca* OCA (Organisation Classification Association) file.

6.2.1 Calibration frames

- **Direct standard star observation:**
  DO category: `CAL_DIR`
  Processed by: `crires_spec_jitter`

  Classification keywords:
  ```
  DPR CATG = CALIB
  DPR TYPE = STD
  DPR TECH = SPECTRUM,DIRECT,OTHER
  DET NCORRS NAME = FowlerNsamp
  ```
  Association keywords:

- **Direct observation of standard star in jitter mode:**
  DO category: `CAL_DIR_JIT`
  Processed by: `crires_spec_jitter`

  Classification keywords:
  ```
  DPR CATG = CALIB
  DPR TYPE = STD
  DPR TECH = SPECTRUM,DIRECT,JITTER
  DET NCORRS NAME = FowlerNsamp
  ```
  Association keywords:

- **Nodding observation of standard star:**
  DO category: `CAL_NOD`
  Processed by: `crires_spec_jitter`

  Classification keywords:
  ```
  DPR CATG = CALIB
  DPR TYPE = STD
  DPR TECH = SPECTRUM,NODDING,OTHER
  DET NCORRS NAME = FowlerNsamp
  ```
  Association keywords:

- **Nodding observation of standard star in windowing mode:**
  DO category: `CAL_NOD_WIN`
  Processed by: `crires_win_jitter`

  Classification keywords:
  ```
  DPR CATG = CALIB
  ```
  Association keywords:

```
DPR TYPE = STD
DPR TECH = SPECTRUM,NODDING,OTHER
DET NCORRS NAME = FowlerNsampGRstWin
```

- **Nodding observation of a standard star in jitter mode:**

  DO category: `CAL_NOD_JIT`
  Processed by: `crires_spec_jitter`

  Classification keywords: Association keywords:

```
DPR CATG = CALIB
DPR TYPE = STD
DPR TECH = SPECTRUM,NODDING,JITTER
DET NCORRS NAME = FowlerNsamp
```

- **Nodding observation of a standard star in jitter mode in windowing mode:**

  DO category: `CAL_NOD_JIT_WIN`
  Processed by: `crires_spec_jitter`

  Classification keywords: Association keywords:

```
DPR CATG = CALIB
DPR TYPE = STD
DPR TECH = SPECTRUM,NODDING,JITTER
DET NCORRS NAME = FowlerNsampGRstWin
```

- **Lamp image for the DETMON linearity-gain calibration:**

  DO category: `ON_RAW`
  Processed by: `detmon_ir_lg`

  Classification keywords: Association keywords:

```
DPR CATG = CALIB
DPR TYPE = FLAT,LAMP,DETCHECK
DPR TECH = LINEARITY,LAMP
```

- **Dark image for the DETMON linearity-gain calibration:**

  DO category: `OFF_RAW`
  Processed by: `detmon_ir_lg`

  Classification keywords: Association keywords:

```
DPR CATG = CALIB
DPR TYPE = DARK,DETCHECK
DPR TECH = LINEARITY,LAMP
```

- **Dark image:**

  DO category: `CAL_DARK`
  Processed by: `crires_spec_dark`

Classification keywords:
```
DPR CATG = CALIB
DPR TYPE = DARK
DPR TECH = IMAGE
DET NCORRS NAME = FowlerNsamp
```
Association keywords:

- **Dark image in windowing mode:**

  DO category: `CAL_DARK_WIN`
  Processed by: `crires_win_dark`

  Classification keywords:
  ```
  DPR CATG = CALIB
  DPR TYPE = DARK
  DPR TECH = IMAGE
  DET NCORRS NAME = FowlerNsampGRstWin
  ```
  Association keywords:

- **Flat image:**

  DO category: `CAL_FLAT`
  Processed by: `crires_spec_flat`

  Classification keywords:
  ```
  DPR CATG = CALIB
  DPR TYPE = FLAT
  DPR TECH = SPECTRUM
  DET NCORRS NAME = FowlerNsamp
  ```
  Association keywords:

- **Flat image in windowing mode:**

  DO category: `CAL_FLAT_WIN`
  Processed by: `crires_win_flat`

  Classification keywords:
  ```
  DPR CATG = CALIB
  DPR TYPE = FLAT
  DPR TECH = SPECTRUM
  DET NCORRS NAME = FowlerNsampGRstWin
  ```
  Association keywords:

- **Wavelength calibration with sky:**

  DO category: `CAL_WLSKY`
  Processed by: `crires_spec_wavecal`

  Classification keywords:
  ```
  DPR CATG = CALIB
  DPR TYPE = WAVE,SKY
  DPR TECH = SPECTRUM
  DET NCORRS NAME = FowlerNsamp
  ```
  Association keywords:

- **Wavelength calibration with a lamp:**

  DO category: `CAL_WLLAMP`
  Processed by: `crires_spec_wavecal`

Classification keywords:
```
DPR CATG = CALIB
DPR TYPE = WAVE,LAMP
DPR TECH = SPECTRUM
DET NCORRS NAME = FowlerNsamp
```
Association keywords:

- **Wavelength calibration with an absorption cell:**
  DO category: `CAL_WLABS`
  Processed by: `crires_spec_wavecal`

  Classification keywords:
  ```
  DPR CATG = CALIB
  DPR TYPE = WAVE,ABSORPTION-CELL
  DPR TECH = SPECTRUM
  DET NCORRS NAME = FowlerNsamp
  ```
  Association keywords:

- **Model configuration file refining:**
  DO category: `MODEL_REFINE_RAW`
  Processed by: `crires_model_refine`

  Classification keywords:
  ```
  DPR CATG = TECHNICAL
  DPR TYPE = WAVE,LAMP
  DPR TECH = SPECTRUM
  DET NCORRS NAME = FowlerNsamp
  ```
  Association keywords:

6.2.2 Science frames

- **Direct observation:**
  DO category: `OBS_DIR`
  Processed by: `crires_spec_jitter`

  Classification keywords:
  ```
  DPR CATG = SCIENCE
  DPR TYPE = OBJECT
  DPR TECH = SPECTRUM,DIRECT,OTHER
  DET NCORRS NAME = FowlerNsamp
  ```
  Association keywords:

- **Direct observation in jitter mode:**
  DO category: `OBS_DIR_JIT`
  Processed by: `crires_spec_jitter`

  Classification keywords:
  ```
  DPR CATG = SCIENCE
  DPR TYPE = OBJECT
  DPR TECH = SPECTRUM,DIRECT,JITTER
  DET NCORRS NAME = FowlerNsamp
  ```
  Association keywords:

- **Object observation:**
  DO category: `OBS_OBJECT`
  Processed by: `crires_spec_jitter`

  Classification keywords:
  `DPR CATG = SCIENCE`
  `DPR TYPE = OBJECT`
  `DPR TECH = SPECTRUM,NODDING`
  `DET NCORRS NAME = FowlerNsamp`

  Association keywords:

- **Sky observation:**
  DO category: `OBS_SKY`
  Processed by: `crires_spec_jitter`

  Classification keywords:
  `DPR CATG = SCIENCE`
  `DPR TYPE = SKY`
  `DPR TECH = SPECTRUM,NODDING`
  `DET NCORRS NAME = FowlerNsamp`

  Association keywords:

- **Object observation in windowing mode:**
  DO category: `OBS_OBJECT_WIN`
  Processed by: `crires_win_jitter`

  Classification keywords:
  `DPR CATG = SCIENCE`
  `DPR TYPE = OBJECT`
  `DPR TECH = SPECTRUM,NODDING`
  `DET NCORRS NAME = FowlerNsampGRstWin`

  Association keywords:

- **Sky observation in windowing mode:**
  DO category: `OBS_SKY_WIN`
  Processed by: `crires_win_jitter`

  Classification keywords:
  `DPR CATG = SCIENCE`
  `DPR TYPE = SKY`
  `DPR TECH = SPECTRUM,NODDING`
  `DET NCORRS NAME = FowlerNsampGRstWin`

  Association keywords:

- **Generic Offset observation:**
  DO category: `OBS_GENERIC_OBJECT`
  Processed by: `crires_spec_jitter`

  Classification keywords:
  `DPR CATG = SCIENCE`
  `DPR TYPE = OBJECT`
  `DPR TECH = SPECTRUM,GENERIC`
  `DET NCORRS NAME = FowlerNsamp`

  Association keywords:

- **Generic Offset Sky observation:**
  DO category: `OBS_GENERIC_SKY`
  Processed by: `crires_spec_jitter`

  Classification keywords: Association keywords:
  ```
  DPR CATG = SCIENCE
  DPR TYPE = SKY
  DPR TECH = SPECTRUM,GENERIC
  DET NCORRS NAME = FowlerNsamp
  ```

- **Nodding observation:**
  DO category: `OBS_NOD`
  Processed by: `crires_spec_jitter`

  Classification keywords: Association keywords:
  ```
  DPR CATG = SCIENCE
  DPR TYPE = OBJECT
  DPR TECH = SPECTRUM,NODDING,OTHER
  DET NCORRS NAME = FowlerNsamp
  ```

- **Nodding observation in windowing mode:**
  DO category: `OBS_NOD_WIN`
  Processed by: `crires_win_jitter`

  Classification keywords: Association keywords:
  ```
  DPR CATG = SCIENCE
  DPR TYPE = OBJECT
  DPR TECH = SPECTRUM,NODDING,OTHER
  DET NCORRS NAME = FowlerNsampGRstWin
  ```

- **Nodding observation in jitter mode:**
  DO category: `OBS_NOD_JIT`
  Processed by: `crires_spec_jitter`

  Classification keywords: Association keywords:
  ```
  DPR CATG = SCIENCE
  DPR TYPE = OBJECT
  DPR TECH = SPECTRUM,NODDING,JITTER
  DET NCORRS NAME = FowlerNsamp
  ```

- **Nodding observation in jitter mode in windowing mode:**
  DO category: `OBS_NOD_JIT_WIN`
  Processed by: `crires_win_jitter`

  Classification keywords: Association keywords:
  ```
  DPR CATG = SCIENCE
  DPR TYPE = OBJECT
  DPR TECH = SPECTRUM,NODDING,JITTER
  DET NCORRS NAME = FowlerNsampGRstWin
  ```

- **Spectro-Astrometry:**
  DO category: `ASTRO_NOD_JIT`
  Processed by: `crires_spec_astro`

  Classification keywords: Association keywords:
  ```
  DPR CATG = SCIENCE
  DPR TYPE = OBJECT
  DPR TECH = SPECTRUM,NODDING,JITTER,ASTROMETRY
  DET NCORRS NAME = FowlerNsamp
  ```

7 Static Calibration Data

For CRIRES, the static calibration data are the photospheric standard star models, the OH, N2O, Thorium/Argon and HITRAN lines, the Y positions for the ThAr calibration and the configuration file for the model.

All these catalogs can be found in the CRIRES pipeline source distribution as ASCII files in crirep/catalogs:

```
ls crirep/catalogs/
model_conf.txt    n2o.txt                 stdstar  thar.txt
hitran_emiss.txt  model_refine_conf.txt   oh.txt   thar_positions.txt

ls crirep/catalogs/stdstar/
BD28D4211.txt  GD108.txt  HIP102497.txt  HR4468.txt  HR7596.txt  HZ2.txt
               GD153.txt  HIP104139.txt  HR4963.txt  HR7950.txt  HZ43.txt
FEIGE67.txt    GD50.txt   HR1544.txt     HR5501.txt  HR8634.txt  NGC7293.txt
G93-48.txt     GD71.txt   HR3454.txt     HR718.txt   HR9087.txt
```

Those calibration tables are also part of the distribution in FITS format. But if they need to be updated (add a new standard, add new emission lines,...) by the user, they can be re-generated from these ASCII catalogs by using the following utilities:

- crires_util_genlines : Generate spectrum calibration FITS tables
- crires_util_genstd : Generate standard star FITS tables
- crires_util_genconfig : Generate model configuration FITS tables
- crires_util_genypos : Generate Y Positions FITS table

For example, the following:

```
$ more IN
/home/yjung/crirep/catalogs/stdstar/BD28D4211.txt STDSTAR
/home/yjung/crirep/catalogs/stdstar/FEIGE67.txt STDSTAR
/home/yjung/crirep/catalogs/stdstar/G93-48.txt STDSTAR
/home/yjung/crirep/catalogs/stdstar/GD108.txt STDSTAR
/home/yjung/crirep/catalogs/stdstar/GD153.txt STDSTAR
/home/yjung/crirep/catalogs/stdstar/GD50.txt STDSTAR
/home/yjung/crirep/catalogs/stdstar/GD71.txt STDSTAR
/home/yjung/crirep/catalogs/stdstar/HIP102497.txt STDSTAR
/home/yjung/crirep/catalogs/stdstar/HIP104139.txt STDSTAR
/home/yjung/crirep/catalogs/stdstar/HR1544.txt STDSTAR
/home/yjung/crirep/catalogs/stdstar/HR3454.txt STDSTAR
```

```
/home/yjung/cripep/catalogs/stdstar/HR4468.txt STDSTAR
/home/yjung/cripep/catalogs/stdstar/HR4963.txt STDSTAR
/home/yjung/cripep/catalogs/stdstar/HR5501.txt STDSTAR
/home/yjung/cripep/catalogs/stdstar/HR718.txt STDSTAR
/home/yjung/cripep/catalogs/stdstar/HR7596.txt STDSTAR
/home/yjung/cripep/catalogs/stdstar/HR7950.txt STDSTAR
/home/yjung/cripep/catalogs/stdstar/HR8634.txt STDSTAR
/home/yjung/cripep/catalogs/stdstar/HR9087.txt STDSTAR
/home/yjung/cripep/catalogs/stdstar/HZ2.txt STDSTAR
/home/yjung/cripep/catalogs/stdstar/HZ43.txt STDSTAR
/home/yjung/cripep/catalogs/stdstar/NGC7293.txt STDSTAR

esorex crires_util_genstd IN
```

will create the static calibration FITS file (PRO.CATG=CALPRO_STD_PHOTOFLUX) needed by *crires_spec_jitter* for standard star observations.

7.1 Standard Star Photosphric flux

This file contains for each of the stars a model of a high resolution photospheric flux of the star (175000 samples from 950 to 5500 nanometers).

Figure 7.1.1 shows an example of such a photospheric flux.

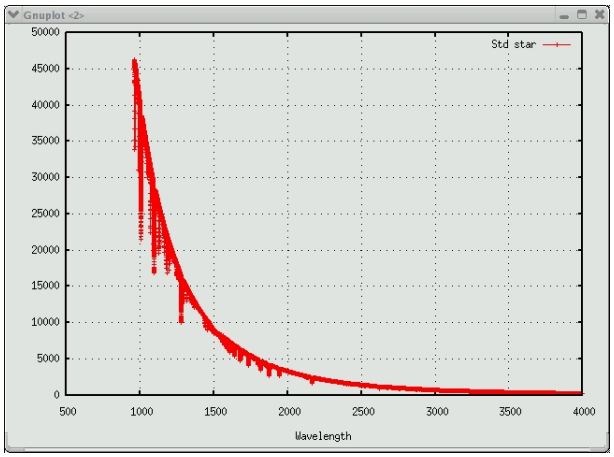

Figure 7.1.1: *Photospheric flux of the star HIP81007.*

7.2 Emission lines

The CRIRES pipeline also needs the emission lines of the atmosphere, of the Thorium and of the Argon gas, as well as the N2O. These are used for the wavelength calibration to generate theoretical signals used to correlate

with the observed ones (OH lines for real observations, N2O, Thorium and Argon for calibration lamps/gas cells).

Figure 7.2.1 shows some lines catalogs used.

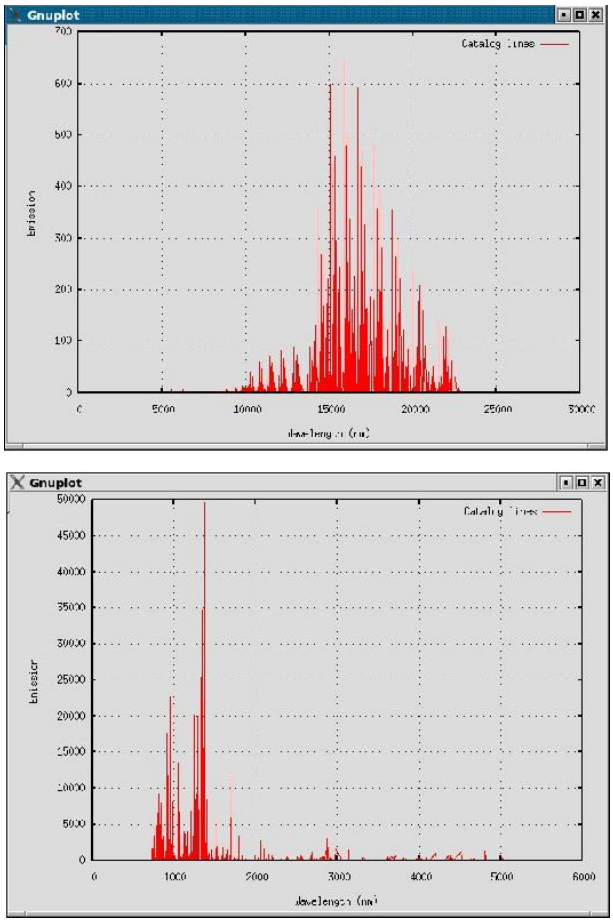

Figure 7.2.1: *OH and Thorium/Argon from top to bottom.*

7.3 Y Positions for ThAr calibration

The *crires_spec_wavecal* recipe needs to be provided with the positions along the slit where to compute the various wavelength calibrations.

```
$ more IN
/home/yjung/cripep/catalogs/thar_positions.txt YPOS

esorex crires_util_genypos IN
```

will create the static calibration FITS file (PRO.CATG=CALPRO_THAR_POS) needed by *crires_spec_wavecal*.

7.4 Model configuration files

The model configuration files are table containing 119 parameters needed by the model to run. For each parameter, it contains the best guess value, a lower and upper limit to define the search space, and a flag defining if the parameter needs to be refined or not.

Appendix B gives the list of the configuration file parameters.

There are 2 such files:

- model_conf.fits (PRO.CATG = CALPRO_MODEL_CONFIG)

  This file is generated from crirep/catalogs/model_conf.txt using crires_util_genconfig with the option –mode=2.

  This file is the common configuration used to compute the wavelength calibration with the model in many different recipes or utilities.

- model_refine_conf.fits (PRO.CATG = CALPRO_MODEL_REFINE_CONFIG)

  This file is generated from crirep/catalogs/model_refine_conf.txt using crires_util_genconfig with the option –mode=1.

  This file has to be used as input to *crires_model_refine* to create a optmized model configuration file.

8 Data Reduction

The data reduction is strongly connected with the observation strategy which in turn depends on the nature of the observed object and the science goals of the observations. The pipeline only treats the most general cases and the user is stronly recommended to read the CRIRES user manual for detailed discussion of the observing strategies and their optimisations.

8.1 Data reduction overview

The main observation technique is nodding. The spectrum is moved along the slit and acquired in two main positions. With successive differences between the two main nodding positions, we get rid of the sky. Nodding is usually combined with small random jitters along the slit, around each nodding position. In case of nodding observations, it is always possible to generate a combined image using all frames, another one created exclusively from nodding A positions, and another one with nodding B positions.

Another observation technique is jittering. In this case, the object is simply moved along the slit using scattered offsets. In this case, sky images are expected as the sky cannot be subtracted with differences like in the nodding mode. In this case, after sky subtraction, the images are shifted and combined together.

A third technique is *generic offset* where the object is moved orthogonally to the slit. Also in this case, some sky images are provided in the data set.

In all cases, the offsets from the header are used as a starting point and are refined, usually using the brightest object in the sky-corrected images.

The bad pixels, the dark, the detector linearity coefficients and the flat field can always be provided for an initial correction of the input images.

Once the combined image is created, the spectrum is extracted. The brightest spectrum is automatically detected and extracted, but options allow the user to specify a zone in the image if he wants a specific spectrum to be extracted.

The wavelength calibration is done using either OH lines, ThAr lines, a HITRAN model, or the instrument physical model. A robust cross-correlation technique is used to align the observed spectra with some synthetic spectra generated from embedded catalogs.

Finally, in the case of standard stars observations, the conversion factor, sensitivity, throughput are derived using the star photospheric flux obtained from a catalog.

8.2 Reduction Cascade

To be able to reduce science data one needs to use raw, product data and pipeline recipes in a given sequence which provides all the input necessary to each pipeline recipe. We call this sequence a data reduction cascade.

Calibration data products can be generated from raw data using the pipeline recipes. Alternatively the user may use calibration products obtained from the ESO archive or from the ESO Data Flow Operation department.

The reduction cascade is described in figure 8.2.1.

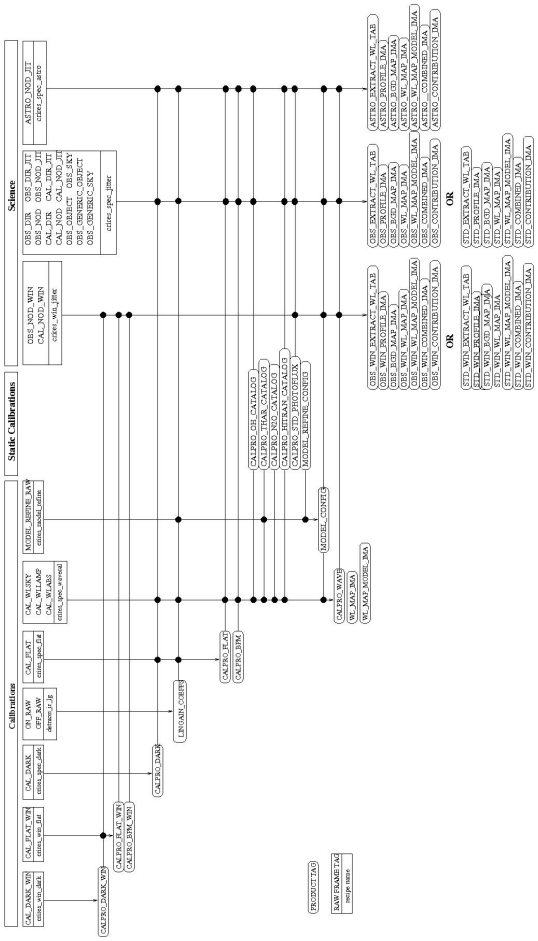

Figure 8.2.1: *CRIRES Association Map.*

8.3 Utilities usage

The recipes are typically run by the automatic pipeline, but can also be run by the users.

In addition to the usual recipes (crires_spec_xxx or crires_win_xxx or crires_model_xxx), a series of utilities (crires_util_xxx) are provided in the pipeline distribution. These utilities can be used to apply smaller data reduction steps than the ones achieved by the recipes and are probably more convenient to follow step by step the data reduction flow for a given data set. They allow the user to, at each step, fine tune the options to improve the intermediate products and final products quality.

They are easy to use and call exactly the same functionalities as the recipes. Use *esorex –man-page* to get the description of their interfaces.

Figure 8.3.1 gives a global overview of their usage to reduce the CRIRES data.

The following section gives a data reduction example to illustrate those utilities usage.

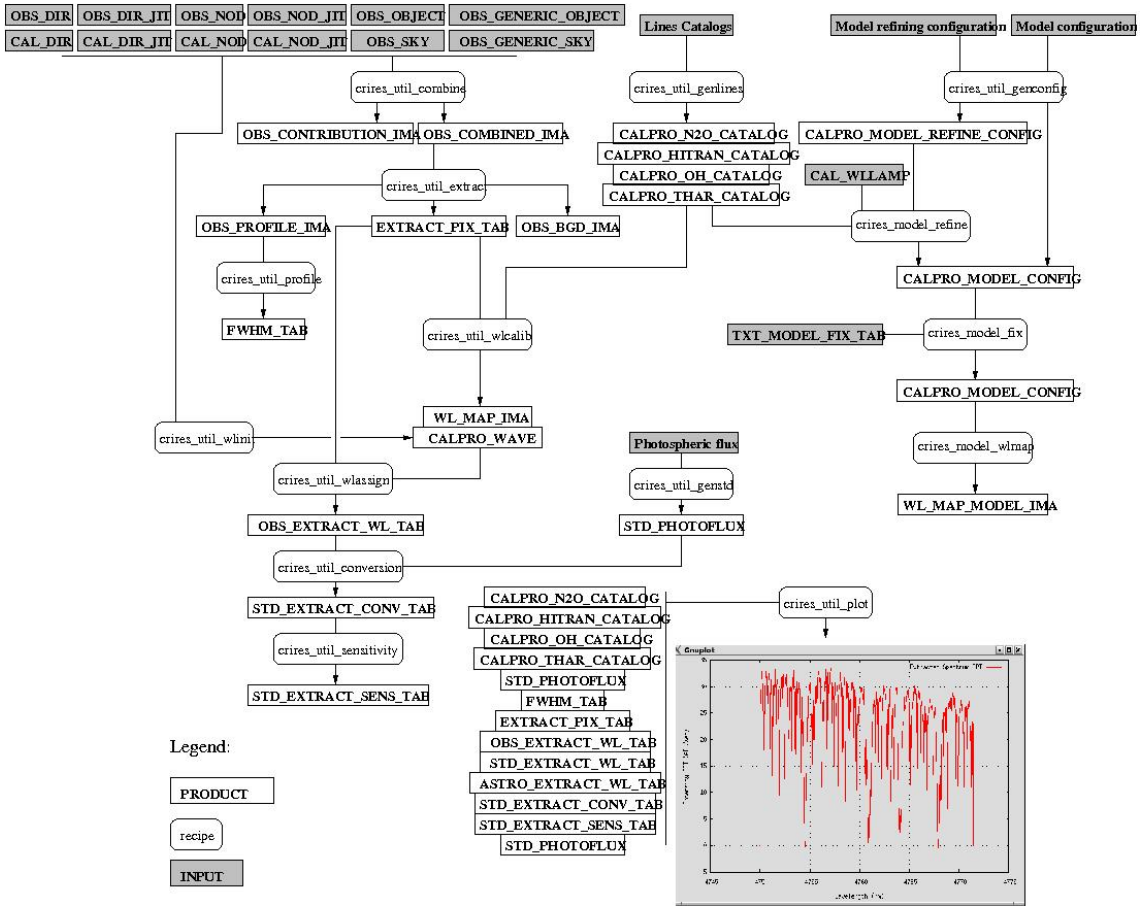

Figure 8.3.1: *CRIRES pipeline utilities.*

8.4 Data Reduction Example using the Utilities

A standard star is observed in nodding mode. We want to flatfield the data, correct the bad pixels, recombine the images together, extract the spectrum from the image combined from the only A positions, from the image combined from the only B nodding positions and from the total combined image. We want then to compare the spectra to see the resolution loss due to the full combination, and to expose the wavelength shift (due to the distortion) between the 2 nodding positions. Then, we calibrate the wavelengths, i.e we compute the pixels-wavelength relation using a ThAr lines image. Finally, we proceed with the sensitivity/conversion factor/throughput computation.

In order to do this, we need:

- The raw files of a dark exposure
- The raw files of the twillight flat
- The raw files for the standard star nodding observation

- A ThAr lamp exposure with the same grism settings as the ones used for the observation
- The ThAr lines catalog (/crire-calib/spec/lines_thar.fits)
- The HITRAN model catalog (/crire-calib/spec/lines_hitran.fits)
- The OH lines catalog (/crire-calib/spec/lines_oh.fits)
- The standard stars photospheric flux catalog (/crire-calib/spec/stdstars.fits)

The 4 last items are static calibration files that are provided with the pipeline distribution (see section 7).

As a general rule, *esorex –recipes* gives the list of available recipes, and *esorex –man-page recipe-name* is very useful to have a quick look at the required inputs, the available parameters and their default values, and at the products of that specified recipe.

8.4.1 Dark computation

If input_dark.txt is the list of raw files of a dark exposure like:

```
CRIRE.2010-12-21T15:15:48.701.fits  CAL_DARK
CRIRE.2010-12-21T15:16:11.067.fits  CAL_DARK
CRIRE.2010-12-21T15:16:33.447.fits  CAL_DARK
```

esorex crires_spec_dark input_dark.txt creates the master dark crires_spec_dark.fits.

8.4.2 Flat field and Bad Pixels

If input_flat.txt is the list of raw files from a flat field template, with additionally the master dark like:

```
CRIRE.2010-12-21T13:58:42.936.fits  CAL_FLAT
CRIRE.2010-12-21T14:00:18.632.fits  CAL_FLAT
CRIRE.2010-12-21T14:01:54.367.fits  CAL_FLAT
crires_spec_dark.fits CALPRO_DARK
```

esorex crires_spec_flat input_flat.txt creates the master flat crires_spec_flat_set01.fits and the bad pixels map crires_spec_flat_set01_bpm.fits

Note that the dark has to have the same DIT as the flat.

8.4.3 Combined Images

In order to create the compined image the file input_combine.txt is created with:

```
CRIRE.2010-12-17T05:44:29.218.fits  OBS_NOD_JIT
CRIRE.2010-12-17T05:44:49.899.fits  OBS_NOD_JIT
CRIRE.2010-12-17T05:45:21.750.fits  OBS_NOD_JIT
CRIRE.2010-12-17T05:45:42.432.fits  OBS_NOD_JIT
CRIRE.2010-12-17T05:46:03.149.fits  OBS_NOD_JIT
CRIRE.2010-12-17T05:46:23.871.fits  OBS_NOD_JIT
CRIRE.2010-12-17T05:46:54.492.fits  OBS_NOD_JIT
CRIRE.2010-12-17T05:47:15.211.fits  OBS_NOD_JIT
crires_spec_flat_set01_bpm.fits     CALPRO_BPM
crires_spec_flat_set01.fits         CALPRO_FLAT
```

and the crires_util_combine is used like this:

esorex crires_util_combine –onlyA –onlyB input_combine.txt

The onlyA (resp. onlyB) option is used to additionally create the combined image using exclusively nodding A (resp. B) position images. The image is called crires_util_combine_comb_noddedA.fits (resp. crires_util_combine_comb_
The combined image created out of all images is crires_util_combine_comb.fits. The extension 1 combined images are shown on figure 8.4.1.

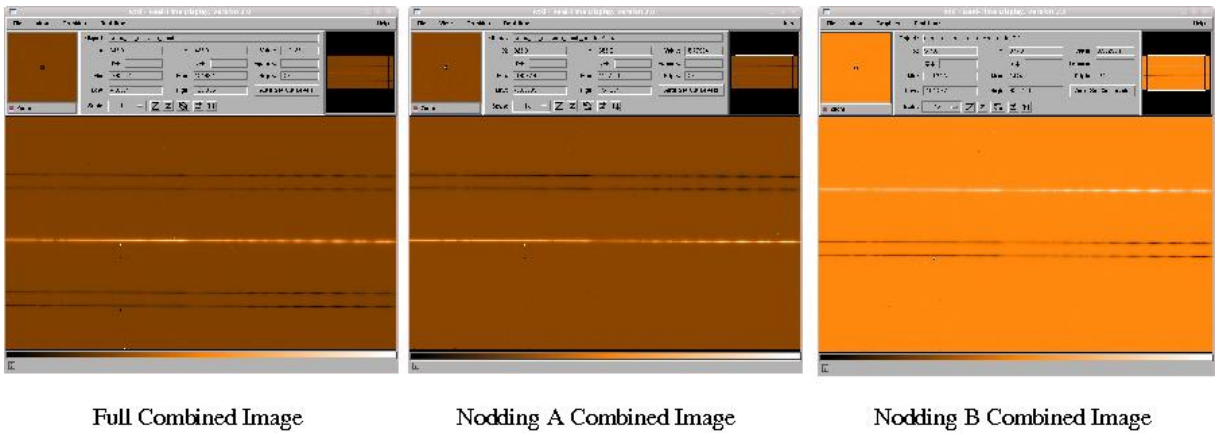

Figure 8.4.1: *Combined images.*

8.4.4 Spectra extraction

The spectrum is then extracted from the 3 combined images.

esorex crires_util_extract crires_util_combine_comb.fits crires_util_combine_contrib.fits

mv crires_util_extract_extracted.fits extracted_all.fits

esorex crires_util_extract crires_util_combine_comb_noddedA.fits crires_util_combine_contrib_noddedA.fits

mv crires_util_extract_extracted.fits extracted_A.fits

esorex crires_util_extract crires_util_combine_comb_noddedB.fits crires_util_combine_contrib_noddedB.fits

mv crires_util_extract_extracted.fits extracted_B.fits

The spectra can be visualised with crires_util_plot : *esorex crires_util_plot –display=1 extracted_A.fits*

would display the spectrum as shown in Figure 8.4.2.

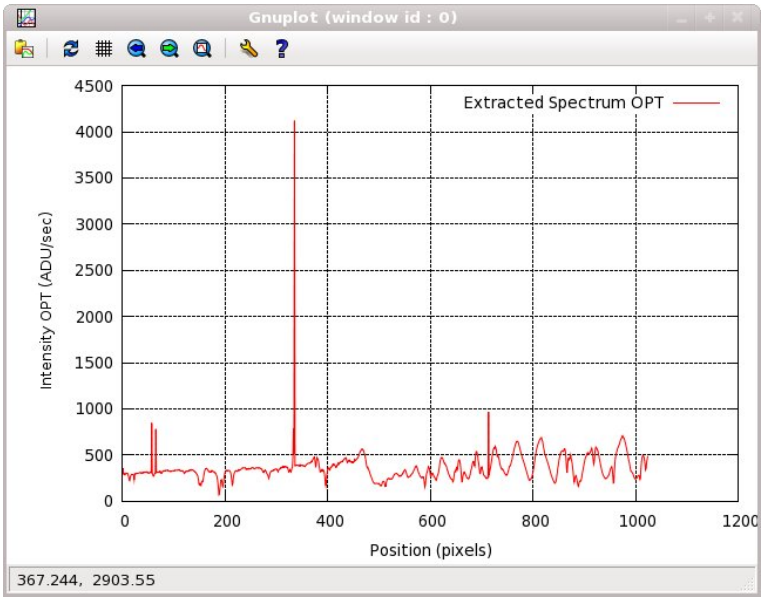

Figure 8.4.2: *Extracted Spectrum.*

We want to observe the degradation of the resolution when the 2 nodding positions are used for the combination. Plotting the first extension of the two extracted spectra around pixel 800 can be done via:

esorex crires_util_plot –display=1 –pmin=790 –pmax=850 extracted_all.fits extracted_A.fits

As shown in Figure 8.4.3, the FWHM in the total combined image is 23.1 pixels, the one for the only A combined image is 21.9 pixels.

The wavelength shift between the 2 nodding positions is due to the distortion and can be exposed with (see Figure 8.4.4) :

esorex crires_util_plot –display=1 –pmin=790 –pmax=850 –adjust extracted_A.fits extracted_B.fits

The approximately 1 pixel shift shows the importance of calibrating in wavelength at the proper Y position.

8.4.5 Wavelength Calibration

As we want to calibrate the nodding A extracted spectrum (which has been extracted at position y=238), we are going to perform the wavelength calibration from that position (See Figure 8.4.5).

The ThAr calibration image is using several fibres along the slit. We are extracting the signal from the fibre close to y=238. This one is situated at y=237 pixels:

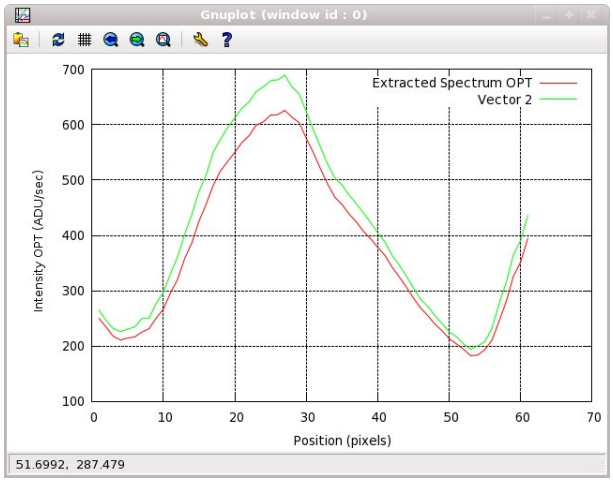

Figure 8.4.3: *Expose the resolution loss when combining all images.*

esorex crires_util_extract –y_pos_c1="237" –mode=2 –display=1 CRIRE.2010-12-22T08:37:55.832.fits

This creates crires_util_extract_extracted.fits which holds the ThAr spectrum extracted around position Y=237 pixels. Using this extracted spectrum, and the ThAr catalog, we can perform the wavelength calibration using:

esorex crires_util_wlcalib crires_util_extract_extracted.fits /crire-calib/spec/lines_thar.fits

This creates crires_util_wlcalib_tab.fits which contains for each detector the wavelength-pixels relations computed along the slit (the so-called wavelength calibration). In our example, only 1 relation was computed. At this stage, we only need to apply this calibration to our previously extracted spectrum with:

esorex crires_util_wlassign extracted_A.fits crires_util_wlcalib_tab.fits

This creates crires_util_wlassign.fits which hold now the extracted spectrum in wavelength, as you can see from the plot generated by the following command (See Figure 8.4.6) :

esorex crires_util_plot –display=1 crires_util_wlassign.fits

8.4.6 Sensitivity / Conversion factor / Throughput

In order to compute the conversion factor and the throughput:

esorex crires_util_conversion crires_util_wlassign.fits /crire-calib/spec/stdstars.fits

This creates crires_util_conversion.fits.

In order to compute the sensitivity:

esorex crires_util_sensitivity crires_util_conversion.fits

This creates crires_util_sensitivity.fits.

All these products can be plotted with e.g. (see Figure 8.4.7):

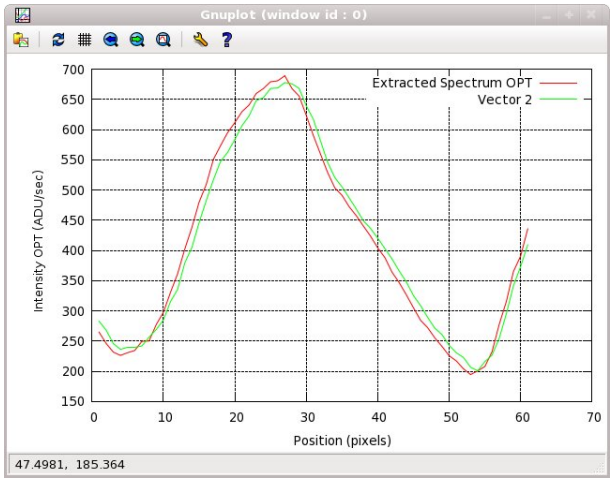

Figure 8.4.4: *Expose the wavelength shift between the 2 nodding positions.*

esorex crires_util_plot –display=3 crires_util_sensitivity.fits

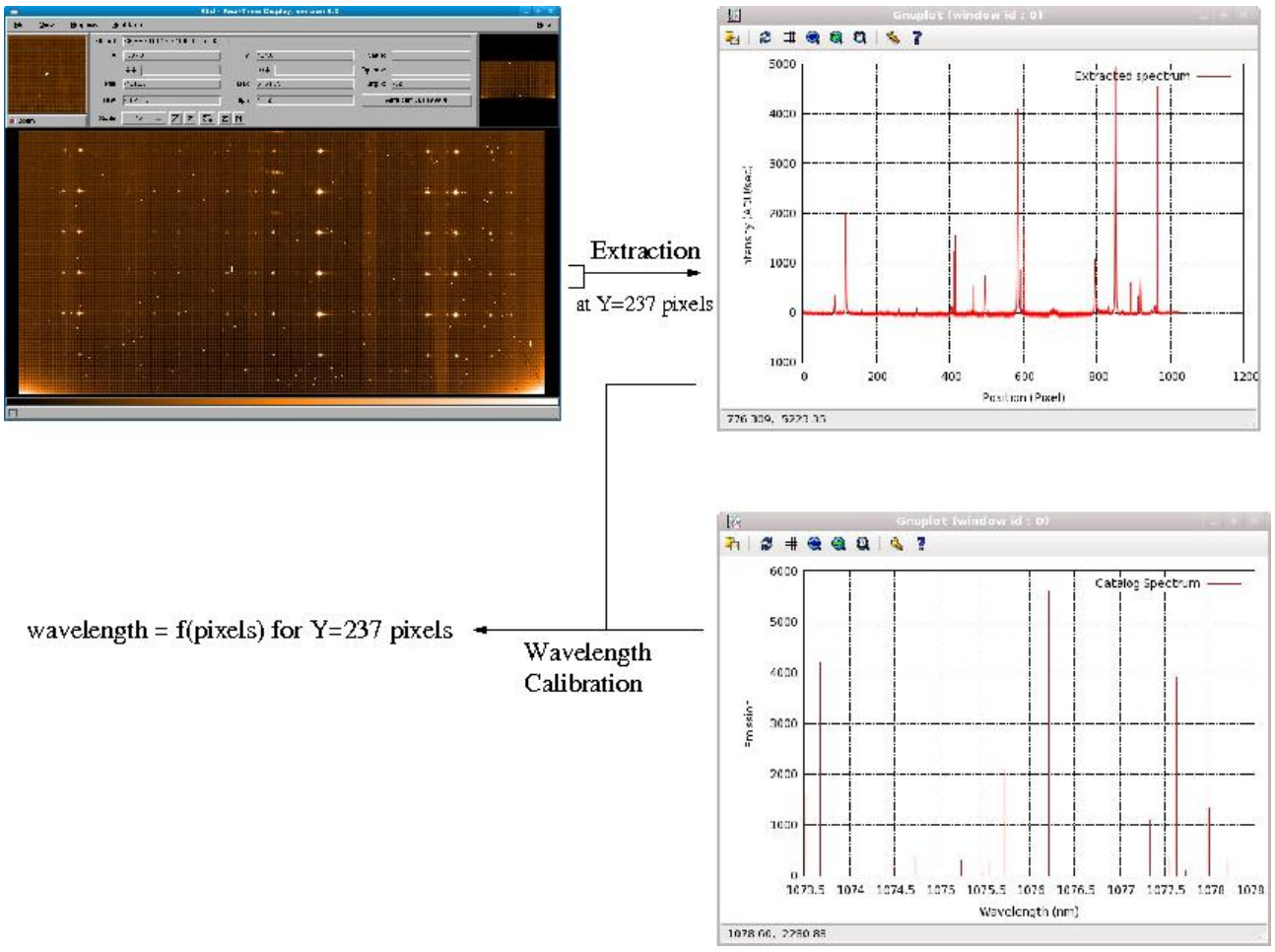

Figure 8.4.5: *Wavelength Calibration for a specific Y position.*

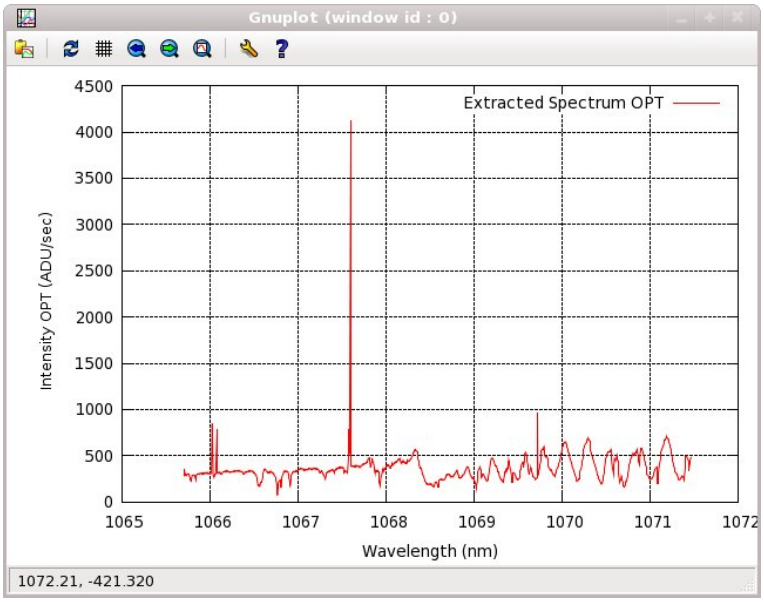

Figure 8.4.6: *Wavelength Calibrated Spectrum.*

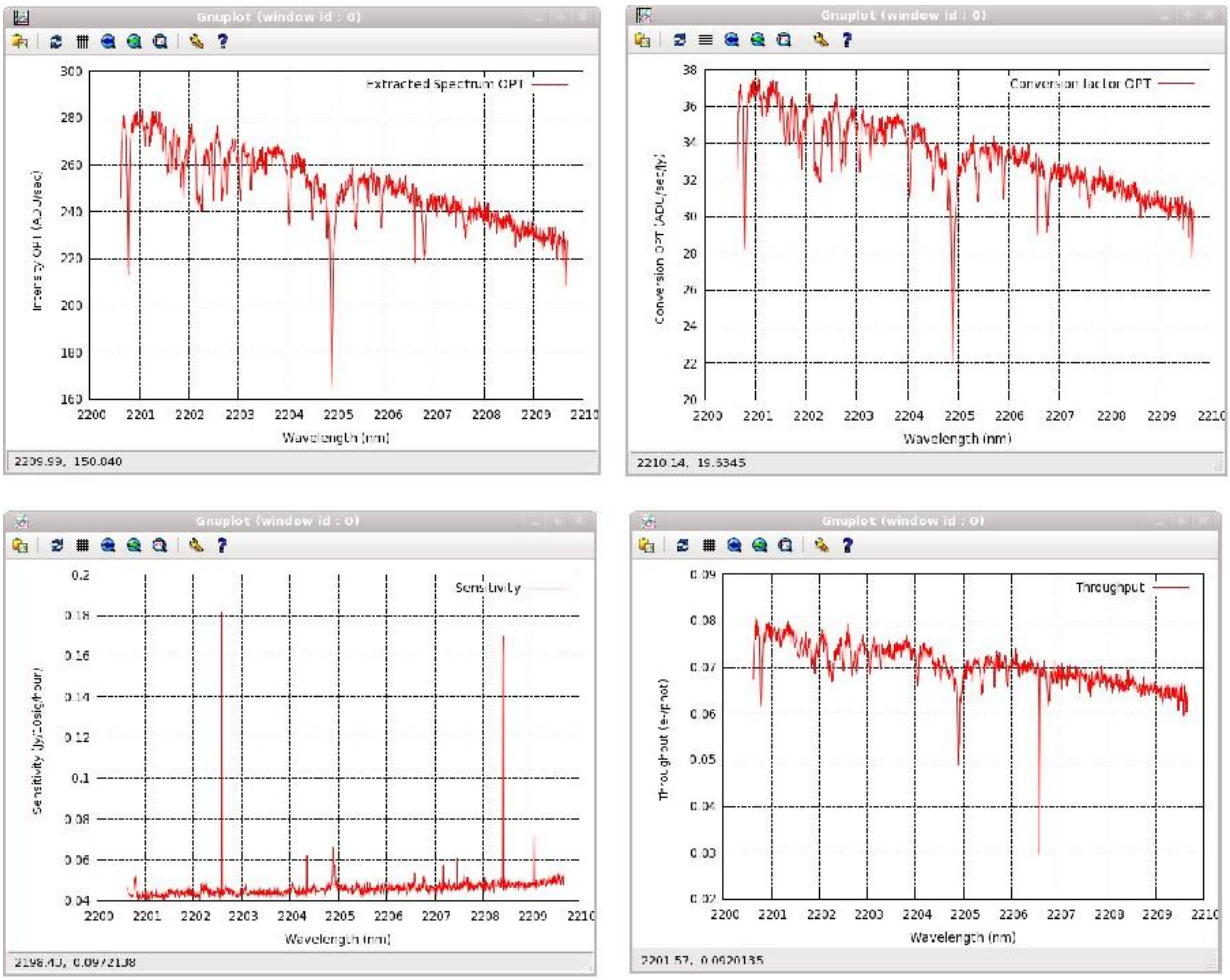

Figure 8.4.7: *Sensitivity / Conversion factor / Throughput.*

9 Pipeline Recipe Interfaces

In this section we provide examples of the required input data for each recipe.

For each recipe, we also provide a list of the pipeline products indicating their default recipe name, the value of the FITS keyword HIERARCH ESO PRO CATG (in short PRO.CATG) and a short description.

For each recipe we also list the input parameters (as they appear in the recipe configuration file), the corresponding aliases for the command line usage, and their default values. Also quality control parameters are listed. Those are stored in relevant pipeline products. More information on instrument quality control can be found on http://www.eso.org/qc

See Section 10 for a description of the algorithms used.

9.1 crires_spec_dark

This recipe creates a master dark image per chip, and computes the Read-Out Noise of the detectors.

9.1.1 Input

This recipe expects input frames classified as CAL_DARK. The input frame set must contain at least 3 frames. The recipes would not run otherwise.

9.1.2 Output

The produced master dark (PRO CATG = CALPRO_DARK) contains 1 extension per chip and is named crires_spec_dark.fits.

9.1.3 Quality control

The QC parameters are computed for each chip. They are stored in the 4 different extensions of the product.

- QC DARKMED: Average of the median values of the different input images for a given chip.
- QC DARKSTDEV: Standard deviation of the median values of the different input images for a given chip.
- QC RON1: Read Out Noise value computed on the difference between the second and the first image for a given chip.
- QC RON2: Read Out Noise value computed on the difference between the third and the second image for a given chip.

9.1.4 Parameters

- nsamples: Number of samples used to compute the RON (default is 100)
- hsize: Half-size of the boxes used to compute the RON (default is 6)

9.2 crires_spec_flat

This recipe generates the spectroscopic flat field for the different chips.

9.2.1 Input

The input frames expected by the recipe are tagged with CAL_FLAT. They are first classified by settings (DIT, reference wavelength, baffle position), and every setting is reduced separately.

It also can accept a dark frame tagged CALPRO_DARK (with the same DIT) and the detector linearity coefficient frame COEFFS_CUBE.

9.2.2 Output

For each setting (xx being the setting number), the master flat named crires_spec_flat_setxx.fits (PRO CATG = CALPRO_FLAT) and the bad pixels map named crires_spec_flat_setxx_bpm.fits (PRO CATG = CALPRO_BPM) are created. A bad pixels map means that the bad pixels are flagged (their value is set to 1).

9.2.3 Quality control

The QC parameters are computed for each chip. They are stored in the 4 different extensions of the product.

- QC NBBAD: The number of bad pixels
- QC FLAT MEAN: The average of median values of the input frames
- QC FLAT STDEV: The standard deviation of median values of the input frames
- QC FLAT FLUX: QC FLAT MEAN divided by the DIT
- QC FLAT MASTER RMS: The standard deviation of the final flat

9.2.4 Parameters

- thresholds: Low and high thresholds to define a ’bad’ pixel (default is "0.5,2.0")
- replace: if set to TRUE, pixels below the low threshold or larger than the high thresold that define a bad pixel are replaced by 1 in the master flat (default is FALSE)

- lines_ratio: Maximum number of bad pixels per line in percent. If a line contains more bad pixels, the whole line pixels are set as good (default is 0.5)
- starty: Y pixel start of the zone used for median normalisation (default is -1 to start from the bottom)
- stopy: Y pixel stop of the zone used for median normalisation (default is -1 to stop at the top)
- coll_meth: Method used to collapse the different flats together. Average (avg), Median (med) and sigma-clipping (sig) methods are supported (default is sig).
- kappa_sigclip: In the case of the sigma-clipping method used for collapsing the flats together, this parameter defines the kappa value (default is 3.0).
- clean_flat: This flag allows to correct the bad pixels directly in the produced flatfield (default is FALSE).

9.3 detmon_ir_lg

The recipe computes the coefficients of a 2nd degree polynomial to correct for the non-linearity of the detectors in a standard way as defined within the DETMON project.

For a full description of the DETMON recipes, please refer to its dedicated documentation ([1]).

9.3.1 Input

This recipe expects input frames classified as ON_RAW or OFF_RAW. There must be the same number of frames with both tags.

9.3.2 Output

The product named detmon_ir_lg_bpm.fits is a bad pixels map (PRO CATG = BP_MAP_NL).

detmon_ir_lg_coeffs_cube.fits is a cube of the coefficients describing the linearity relation (PRO CATG = COEFFS_CUBE).

detmon_ir_lg_gain_table.fits (PRO CATG = GAIN_INFO) and detmon_ir_lg_linearity_table.fits (PRO CATG = DET_LIN_INFO) are tables containing various statistics.

9.3.3 Quality control

The QC parameters are computed for each chip. They are stored in the 4 different extensions of the product.

9.3.4 Parameters

The DETMON recipes are able to reduce data from a large number of instruments. In order to achieve this, the recipe can be fine-tuned using the various parameters. Only a subset of those parameters are relevant for

CRIRES, and only a subset of these must be set to a different value than their default value. These parameters are described here :

- order : Polynomial order for the fit (Linearity)
- ref_level : User reference level
- filter : Upper limit of Median flux to be filtered
- exts : Activate the multi-exts option. Choose -1 to process all extensions

The *detmon_ir_lg* recipe should alway be run on CRIRES data with those default values like:

```
more detmon_ir_lg_crires.rc
detmon.detmon_ir_lg.order=2
detmon.detmon_ir_lg.ref_level=6000
detmon.detmon_ir_lg.filter=12000
detmon.detmon_ir_lg.exts=-1

esorex --recipe-config=detmon_ir_lg_crires.rc detmon_ir_lg sof
```

9.4 crires_spec_wavecal

This recipe computes the wavelength calibration using observations of a ThAr lamp, a halogen lamp seen through a gas cell or sky emission lines.

9.4.1 Input

The input frames must be tagged respectively CAL_WLSKY, CAL_WLLAMP or CAL_WLABS if they are frames obtained from the sky (OH or other sky emission lines), from a lamp (Thorium/Argon lines) or from a gas cell (N2O lines).

As calibrations, the recipe accepts optionally a flat frame (CALPRO_FLAT), a bad pixels map (CALPRO_BPM), a dark frame (CALPRO_DARK), the Y positions for thewavelength calibrations (CALPRO_THAR_POS) and/or the polynomial coefficients for the non-linearity correction (COEFFS_CUBE).

Besides, the catalogs of lines need to be passed depending on the input frames. For CAL_WLSKY inputs, CALPRO_OH_CATALOG and CALPRO_HITRAN_CATALOG catalogs are needed, for CAL_WLLAMP inputs, CALPRO_THAR_CATALOG catalog is needed, and for CAL_WLABS inputs, CALPRO_N2O_CATALOG is needed.

Finally, the configuration file for the model is needed (CALPRO_MODEL_CONFIG) to compute the wavelength map based on the instrument model.

9.4.2 Output

The recipe produces a 4 extensions file with tables containing one wavelength calibration per row (CAL-PRO_WAVE). A wavelength calibration is a polynomial relation wavelength=F(pixel) at a given Y position along the slit.

It also produces 2 wavelength maps, the one obtained by cross-correlation (WL_MAP_IMA) and the one computed by the model (WL_MAP_MODEL_IMA).

Figure 9.4.1 shows one extension of the wavelength solution file.

```
#
# file           crires_spec_wavecal_tab.fits
# extensions     4
# ---------------------------------------------
# XTENSION       2
# Number of columns 7
#
   Coeff_a|   Coeff_b|   Coeff_c|   Coeff_d|   Coeff_e|Y_Position|XCorrelation
   1081.01|0.00526491|         0|         0|         0|        80|          -1
   1081.03|0.00522135|1.74972e-07|         0|         0|       160|     0.57307
   1081.03|0.00531624|-6.50722e-08|         0|         0|       240|     0.28812
   1081.03|0.00530908|-5.7842e-08|         0|         0|       320|    0.433405
   1081.02|0.00531809|-6.45902e-08|         0|         0|       398|    0.631744
   1081.01|0.00526491|         0|         0|         0|       476|          -1
```

Figure 9.4.1: *Example of a wavelength solution.*

9.4.3 Quality control

- QC CENTWL: The computed central wavelength of the detector
- QC DISPWL: The average dispersion of the detector
- QC XCORR: The maximum cross-correlation factor
- QC LINES FLUX: The median of the fluxes of the identified THAR lines
- QC FWHM MED: The median of the FWHMs of the identified THAR lines
- QC RESOL MED: The median of the resolving power of the identified THAR lines

9.4.4 Parameters

- waves: Wavelength at start and end of the detectors (default is to read this from the header)
- display: Chip for which the results are plotted (default is none)
- wl_log: Flag to use the logarithm of the catalog (default is FALSE)
- wl_nolimit: Flag to ignore the WMIN and WMAX values from the header (default is FALSE)
- wl_err: The wavelength error (default is 2.0 nanometer)

- wl_xclimit: The lowest allowed value for XC factor (default is 0.2)
- wl_samples: Number of samples for the wavelength calibration (default is 100)
- y_pos_c1: List of Y positions on CHIP 1 where to perform the wavelength calibrations (default is [] - use hardcoded values)
- y_pos_c2: List of Y positions on CHIP 2 where to perform the wavelength calibrations (default is [] - use hardcoded values)
- y_pos_c3: List of Y positions on CHIP 3 where to perform the wavelength calibrations (default is [] - use hardcoded values)
- y_pos_c4: List of Y positions on CHIP 4 where to perform the wavelength calibrations (default is [] - use hardcoded values)
- y_width: Width in pixels along Y of the zones to extract for the wavelength calibrations (default is 10)
- degree: degree of the polynomial for the solution (default is 2)
- wl_clean: Flag to remove the low freq. part from the spectrum (default is FALSE)

9.5 crires_model_refine

This recipe refines the model configuration file using a lamp exposure, the lamp lines catalog and a first guess model configuration file.

9.5.1 Input

The input frame must be tagged MODEL_REFINE_RAW. It is a lamp exposure.

As calibrations, the recipe needs the lamp lines catalog (CALPRO_THAR_CATALOG), and a first guess model configuration file (CALPRO_MODEL_REFINE_CONFIG).

The recipe accepts optionnally a flat frame (CALPRO_FLAT), a bad pixels map (CALPRO_BPM) and/or the polynomial coefficients for the non-linearity correction (COEFFS_CUBE).

9.5.2 Output

The recipe produces the refined model configuration file named crires_model_refine.fits (PRO CATG = CALPRO_MODEL_CONFIG).

9.5.3 Quality control

No QC parameters.

9.5.4 Parameters

- display: The chip to display (1 -> 4) (default is 0)
- fwhm: The FWHM for the convolution (default is 3)
- sigma: The sigma for the detection (default is 3.0)
- search_box_sz: The size of the search box for lines identification (default is 10)
- min_matches_nb: The minimum number of match requested (default is 3)

9.6 crires_model_fix

This recipe refines the model configuration file using any CRIRES file (for its header), a TXT file with user specified pixel position - wavelength associations and a first guess model configuration file.

9.6.1 Input

The input frame must be tagged MODEL_FIX_RAW. It can be any CRIRES file, and it is used for its configuration defined in its header.

As calibrations, the recipe needs the text file with the pixel-wavelength associations (PRO CATG = CALPRO_MODEL_FIX_TAB). Each line of the text file defines an association, following the syntax:

x y chip wl

where (x,y) is the pixel position on detector number *chip* and *wl* is the associated wavelength.

A first guess model configuration file (PRO CATG = CALPRO_MODEL_REFINE_CONFIG or PRO CATG = CALPRO_MODEL_CONFIG) is also needed.

9.6.2 Output

The recipe produces a model configuration file named crires_model_fix.fits (CALPRO_MODEL_CONFIG).

9.6.3 Quality control

No QC parameters.

9.6.4 Parameters

No parameters

9.7 crires_model_wlmap

This recipe creates a 2D wavelength map using the model.

9.7.1 Input

The input frame must be tagged MODEL_WLMAP_RAW. It can be any CRIRES file, and it is used for its configuration defined in its header.

The recipe needs a first guess model configuration file (PRO CATG = CALPRO_MODEL_REFINE_CONFIG or PRO CATG = CALPRO_MODEL_CONFIG).

9.7.2 Output

The recipe produces a 2D wavelength map named crires_model_wlmap.fits (PRO CATG = WL_MAP_MODEL_IMA).

9.7.3 Quality control

No QC parameters.

9.7.4 Parameters

- order: The order.

9.8 crires_spec_jitter

This recipe is the main observation data reduction tool. It supports direct observation, nodding, jittering, generic offsets and standard stars observations.

9.8.1 Input

Depending on the type of observations, the input raw frames need to be tagged with OBS_DIR for direct observation, OBS_DIR_JIT if some jittering is applied, OBS_NOD in nodding mode, OBS_NOD_JIT if some jittering is applied, and CAL_DIR, CAL_DIR_JIT, CAL_NOD or CAL_NOD_JIT if a standard star is observed. Additionnally, the tags OBS_OBJECT and OBS_SKY are supported if the science observations are mixed with sky observations. Finally, the generic offset data are tagged with OBS_GENERIC_OBJECT and OBS_GENERIC_SKY.

Besides, the recipe optionnally accepts a bad pixel map (CALPRO_BPM), a flatfield (CALPRO_FLAT), a dark (CALPRO_DARK), detector non-linearity coefficients (COEFFS_CUBE) and/or a table containing the wavelength to pixel relation (CALPRO_WAVE).

For the wavelength calibration it also needs the line catalogs (PRO CATG = CALPRO_OH_CATALOG or/and PRO CATG = CALPRO_HITRAN_CATALOG).

If a model configuration file is given (CALPRO_MODEL_CONFIG), the wavelength calibration as given by the model is also calculated.

In case of standard stars observations, it also needs the standard stars photospheric flux catalog (PRO CATG = CALPRO_STD_PHOTOFLUX).

9.8.2 Output

The recipe produces up to 11 files. They all contain 4 extensions (1 by chip).

- The combined image (PRO CATG = OBS_COMBINED_IMA or PRO CATG = STD_COMBINED_IMA) named crires_spec_jitter_comb.fits
- The combined image out of the A nodding position only (PRO CATG = OBS_COMBINED_NA_IMA or PRO CATG = STD_COMBINED_NA_IMA) named crires_spec_jitter_comb_noddedA.fits
- The combined image out of the B nodding position only (PRO CATG = OBS_COMBINED_NB_IMA or PRO CATG = STD_COMBINED_NB_IMA) named crires_spec_jitter_comb_noddedB.fits
- The contribution map (PRO CATG = OBS_CONTRIBUTION_IMA or PRO CATG = STD_CONTRIBUTION_IMA named crires_spec_jitter_contrib.fits
- The contribution map out of the A nodding position only (PRO CATG = OBS_CONTRIBUTION_NA_IMA or PRO CATG = STD_CONTRIBUTION_NA_IMA) named crires_spec_jitter_contrib_noddedA.fits
- The contribution map out of the B nodding position only (PRO CATG = OBS_CONTRIBUTION_NB_IMA or PRO CATG = STD_CONTRIBUTION_NB_IMA) named crires_spec_jitter_contrib_noddedB.fits
- The table with the extracted spectrum (PRO CATG = OBS_EXTRACT_WL_TAB or PRO CATG = STD_EXTRACT_WL_TAB or PRO CATG = STD_EXTRACT_SENS_TAB or PRO CATG = STD_EXTRACT_CO named crires_spec_jitter_extracted.fits
- The profile image (PRO CATG = OBS_PROFILE_IMA or PRO CATG = STD_PROFILE_IMA) named crires_spec_jitter_prof.fits
- The background map (PRO CATG = OBS_BGD_MAP_IMA or PRO CATG = STD_BGD_MAP_IMA) named crires_spec_jitter_bgmap.fits
- The wavelength map computed by cross-correlation (PRO CATG = OBS_WL_MAP_IMA or PRO CATG = STD_WL_MAP_IMA) named crires_spec_jitter_wlmap.fits
- The wavelength map computed by the model (PRO CATG = OBS_WL_MAP_MODEL_IMA or PRO CATG = STD_WL_MAP_MODEL_IMA) named crires_spec_jitter_wlmap_model.fits

Figure 9.8.1 shows one extension of the spectrum extraction table.

```
#
# file          crires_spec_jitter_extracted.fits
# extensions    4
# ---------------------------------------------
# XTENSION      1
# Number of columns 9
#
Extracted_RECT|Extracted_OPT|Error_RECT| Error_OPT|Weight_Norm_RECT|Weight_Norm_OPT|Background_noise|Wavelength|Wavelength_model
            0|            0|         0|         0|              41|         267391|               0|   3529.43|          3529.29
            0|            0|         0|         0|              41|         267391|               0|   3529.45|          3529.31
      80.6781|      79.9142| 0.170413| 0.134075|              41|         5563.8|       0.0203456|   3529.47|          3529.33
      84.2373|      83.8427|  0.16766|  0.13627|              41|        6006.83|       0.0194489|   3529.49|          3529.34
      84.0055|      84.0935| 0.142781| 0.131283|              41|        10409.3|       0.0138104|    3529.5|          3529.36
      83.9681|      84.6279| 0.139562|  0.13089|              41|        11412.4|       0.0129883|   3529.52|          3529.38
      84.8318|      84.8965| 0.161915| 0.135925|              41|        6735.34|       0.0181634|   3529.54|          3529.39
      84.3965|      84.5349| 0.155418| 0.134321|              41|         7677.1|       0.0167693|   3529.55|          3529.41
      84.8035|      84.8484| 0.138974| 0.130736|              41|        11777.5|       0.0127142|   3529.57|          3529.43
      84.8385|      84.6923| 0.158161| 0.134975|              41|        7272.21|        0.017337|   3529.59|          3529.45
      85.0065|       84.849| 0.133814| 0.129186|              41|        14038.7|       0.0112507|   3529.61|          3529.46
      85.1603|      84.8226| 0.184323| 0.140192|              41|        4581.82|       0.0227582|   3529.62|          3529.48
      85.2816|      85.4895| 0.153044|  0.13452|              41|        8174.02|       0.0161282|   3529.64|           3529.5
      84.9787|      84.5825| 0.155164| 0.134306|              41|        7767.55|       0.0166482|   3529.66|          3529.51
      85.1479|      85.0575| 0.154485|  0.13446|              41|        7899.89|       0.0164747|   3529.68|          3529.53
      85.2992|      85.0337|  0.13425| 0.131624|              41|        7913.77|       0.0113295|   3529.69|          3529.55
      85.0725|      84.5673|  0.15437| 0.134131|              41|        7928.67|       0.0164568|   3529.71|          3529.57
       85.139|       85.449| 0.153211| 0.134774|              41|        7928.54|       0.0161829|   3529.73|          3529.58
      84.7096|      85.1503| 0.147786|  0.13381|              41|        7943.17|       0.0149542|   3529.74|           3529.6
       85.274|      85.2125| 0.170783| 0.137147|              41|        7929.66|       0.0200064|   3529.76|          3529.62
      85.5989|      85.3829| 0.141917| 0.133046|              41|        7930.76|       0.0133768|   3529.78|          3529.64
```

Figure 9.8.1: *Example of an extracted spectrum table.*

9.8.3 Quality control

- QC SPECPOS: the spectrum position in pixels
- QC SPECWREC: the spectrum width in rectangular extraction
- QC SPECWOPT: the spectrum width in optimal extraction
- QC SIGNAL MED: the median of the optimally extracted spectrum
- QC S2NMED: the median of the signal to noise of the optimally extracted spectrum
- QC XCORR: the maximal cross correlation factor
- QC CENTWL: the central wavelength of the detector
- QC DISPWL: the average dispersion of the detector
- QC SENSMED: the median of the sensitivity estimates
- QC CONVMED: the median of the conversion factor estimates
- QC THROMED: the median of the throughput estimates
- QC FWHMPIX COMBINED: median FWHM of cross-dispersion profile in the combined frame (in pixels)
- QC FWHMARC COMBINED: median FWHM of cross-dispersion profile in the combined frame (in arcsec)
- QC FWHMPIX PROFILE: median FWHM of the extraction profile (in pixels)
- QC FWHMARC PROFILE: median FWHM of the extraction profile (in arcsec)
- QC FWHM DIFF: QC FWHMPIX PROFILE - QC FWHMPIX COMBINED

9.8.4 Parameters

- display: Chip for which the results are plotted (default is none)
- hor_size: box size for profile computation (default is 31)
- spec_hsize: Spectrum half size (default is 20)
- closing_hs: Closing kernel half size for BGD cleaning (default is 2)
- wl_samples: Number of samples for the wavelength calibration (default is 100)
- y_pos_c1: List of Y positions on CHIP 1 where to perform the wavelength calibrations (default is [] - use spectrum position)
- y_pos_c2: List of Y positions on CHIP 2 where to perform the wavelength calibrations (default is [] - use spectrum position)
- y_pos_c3: List of Y positions on CHIP 3 where to perform the wavelength calibrations (default is [] - use spectrum position)
- y_pos_c4: List of Y positions on CHIP 4 where to perform the wavelength calibrations (default is [] - use spectrum position)
- y_width: Width in pixels along Y of the zones to extract for the wavelength calibrations (default is 10)
- degree: degree of the polynomial for the solution (default is 2)
- kappa: kappa for background removal (default is 1.0)
- clean_rate: Minimum rate of bgd pixels per line allowed (default is 0.5)
- wl_err: The wavelength error (default is 2.0 nanometer)
- wl_xclimit: The lowest allowed value for XC factor (default is 0.2)
- waves: Wavelength at start and end of the detectors (default is to read this from the header)
- rej: Left and right rejections (default is 0,0)
- spec_zone: Spectrum position zone specification (default is -1,-1)
- comb_used: Combined image to be used : NODA / NODB / COMB (default is COMB)
- refine: Flag to refine the offsets (default is TRUE)
- onlyA: Also combine using only nodding A (default is FALSE)
- onlyB: Also combine using only nodding B (default is FALSE)
- blind: Flag to get blindly the offsets (default is FALSE)
- wl_log: Flag to use the logarithm of of the wavelengths listed in the catalog (default is FALSE)
- wl_nolimit: Flag to ignore the WMIN and WMAX values from the header (default is FALSE)
- wl_clean: Flag to remove the low freq. part from the spectrum (default is FALSE)

9.9 crires_spec_astro

This recipe is the spectro-astrometric data reduction tool. The input frames are sorted by groups of different derotator angles. For each of these groups, the same algorithms as for the crires_spec_jitter (see 9.8) recipe is applied.

9.9.1 Input

The input raw frames are tagged ASTRO_NOD_JIT.

Besides, the recipe optionnally accepts a bad pixel map (CALPRO_BPM), a flatfield (CALPRO_FLAT), a dark (CALPRO_DARK), detector non-linearity coefficients (COEFFS_CUBE) and/or a table containing the wavelength to pixel relation (CALPRO_WAVE).

For the wavelength calibration it needs the line catalogs (CALPRO_OH_CATALOG, CALPRO_HITRAN_CATALOG). It also needs the model configuration file to run the model (CALPRO_MODEL_CONFIG).

9.9.2 Output

The recipe produces the same products as crires_spec_jitter for each of the input groups (see 9.8)

9.9.3 Quality control

The QC parameters are the same as crires_spec_jitter (see 9.8).

9.9.4 Parameters

The parameters are the same as in crires_spec_jitter (see 9.8).

9.10 crires_win_dark

This recipe creates a master dark image per chip, and computes the Read-Out Noise of the detectors. It is run on the data produced with the read-out mode *FowlerNsampGRstWin*, so detector 1 and 4 are ignored.

9.10.1 Input

This recipe expects input frames classified as CAL_DARK_WIN. The input frame set must contain at least 3 frames. The recipes would not run otherwise.

9.10.2 Output

The produced master dark (PRO CATG = CALPRO_DARK_WIN) contains 1 extension per chip and is named crires_win_dark.fits.

9.10.3 Quality control

The QC parameters are computed for each chip. They are stored in the 4 different extensions of the product.

- QC DARKMED: Average of the median values of the different input images for a given chip.
- QC DARKSTDEV: Standard deviation of the median values of the different input images for a given chip.
- QC RON1: Read Out Noise value computed on the difference between the second and the first image for a given chip.
- QC RON2: Read Out Noise value computed on the difference between the third and the second image for a given chip.

9.10.4 Parameters

- nsamples: Number of samples used to compute the RON (default is 100)
- hsize: Half-size of the boxes used to compute the RON (default is 6)

9.11 crires_win_flat

This recipe generates the spectroscopic flat field for the different chips for data produced with the read-out mode *FowlerNsampGRstWin*.

9.11.1 Input

The input frames expected by the recipe are tagged with CAL_FLAT_WIN. They are first classified by settings (DIT, reference wavelength, baffle position), and every setting is reduced separately.

It also can accept a dark frame tagged CALPRO_DARK_WIN.

9.11.2 Output

For each setting (xx being the setting number), the master flat named crires_win_flat_setxx.fits (PRO CATG = CALPRO_FLAT_WIN) and the bad pixels map named crires_win_flat_setxx_bpm.fits (PRO CATG = CALPRO_BPM_WIN) are created. A bad pixels map means that the bad pixels are flagged (their value is set to 1).

9.11.3 Quality control

The QC parameters are computed for each chip. They are stored in the 4 different extensions of the product.

- QC NBBAD: The number of bad pixels
- QC FLAT MEAN: The average of median values of the input frames
- QC FLAT STDEV: The standard deviation of median values of the input frames
- QC FLAT FLUX: QC FLAT MEAN divided by the DIT
- QC FLAT MASTER RMS: The standard deviation of the final flat

9.11.4 Parameters

- thresholds: Low and high thresholds to define a 'bad' pixel (default is "0.5,2.0")
- replace: if set to TRUE, pixels below the low threshold or larger than the high thresold that define a bad pixel are replaced by 1 in the master flat (default is FALSE)
- lines_ratio: Maximum number of bad pixels per line in percent. If a line contains more bad pixels, the whole line pixels are set as good (default is 0.5)

9.12 crires_win_jitter

This recipe is the main observation data reduction tool for data taken with the read-out *FowlerNsampGRstWin*. It supports nodding and standard stars observations.

9.12.1 Input

Depending on the type of observations, the input raw frames need to be tagged with OBS_NOD_WIN in nodding mode, OBS_NOD_JIT_WIN if some jittering is applied, and CAL_NOD_WIN or CAL_NOD_JIT_WIN if a standard star is observed. Additionnally, the tags OBS_OBJECT_WIN and OBS_SKY_WIN are supported if the science observations are mixed with sky observations.

Besides, the recipe optionnally accepts a bad pixel map (CALPRO_BPM_WIN), a flatfield (CALPRO_FLAT_WIN), a dark (CALPRO_DARK_WIN) and/or a table containing the wavelength to pixel relation (CALPRO_WAVE).

If a model configuration file is given (CALPRO_MODEL_CONFIG), the wavelength calibration as given by the model is also calculated.

In case of standard stars observations, it also needs the standard stars photospheric flux catalog (PRO CATG = CALPRO_STD_PHOTOFLUX).

9.12.2 Output

The recipe produces up to 11 files. They all contain 4 extensions (1 by chip).

- The combined image (PRO CATG = OBS_WIN_COMBINED_IMA or PRO CATG = STD_WIN_COMBINED_IMA named crires_win_jitter_comb.fits
- The combined image out of the A nodding position only (PRO CATG = OBS_WIN_COMBINED_NA_IMA or PRO CATG = STD_WIN_COMBINED_NA_IMA) named crires_win_jitter_comb_noddedA.fits
- The combined image out of the B nodding position only (PRO CATG = OBS_WIN_COMBINED_NB_IMA or PRO CATG = STD_WIN_COMBINED_NB_IMA) named crires_win_jitter_comb_noddedB.fits
- The contribution map (PRO CATG = OBS_WIN_CONTRIBUTION_IMA or PRO CATG = STD_WIN_CONTRIBU named crires_win_jitter_contrib.fits
- The contribution map out of the A nodding position only (PRO CATG = OBS_WIN_CONTRIBUTION_NA_IMA or PRO CATG = STD_WIN_CONTRIBUTION_NA_IMA) named crires_win_jitter_contrib_noddedA.fits
- The contribution map out of the B nodding position only (PRO CATG = OBS_WIN_CONTRIBUTION_NB_IMA or PRO CATG = STD_WIN_CONTRIBUTION_NB_IMA) named crires_win_jitter_contrib_noddedB.fits
- The table with the extracted spectrum (PRO CATG = OBS_WIN_EXTRACT_WL_TAB or PRO CATG = STD_WIN_EXTRACT_WL_TAB or PRO CATG = STD_WIN_EXTRACT_SENS_TAB or PRO CATG = STD_WIN_EXTRACT_CONV_TAB) named crires_win_jitter_extracted.fits
- The profile image (PRO CATG = OBS_WIN_PROFILE_IMA or PRO CATG = STD_WIN_PROFILE_IMA) named crires_win_jitter_prof.fits
- The background map (PRO CATG = OBS_WIN_BGD_MAP_IMA or PRO CATG = STD_WIN_BGD_MAP_IMA) named crires_win_jitter_bgmap.fits
- The wavelength map computed by cross-correlation (PRO CATG = OBS_WIN_WL_MAP_IMA or PRO CATG = STD_WIN_WL_MAP_IMA) named crires_win_jitter_wlmap.fits
- The wavelength map computed by the model (PRO CATG = OBS_WIN_WL_MAP_MODEL_IMA or PRO CATG = STD_WIN_WL_MAP_MODEL_IMA) named crires_win_jitter_wlmap_model.fits

9.12.3 Quality control

- QC SPECPOS: the spectrum position in pixels
- QC SPECWREC: the spectrum width in rectangular extraction
- QC SPECWOPT: the spectrum width in optimal extraction
- QC SIGNAL MED: the median of the optimally extracted spectrum
- QC S2NMED: the median of the signal to noise of the optimally extracted spectrum

- QC CENTWL: the central wavelength of the detector
- QC DISPWL: the average dispersion of the detector
- QC SENSMED: the median of the sensitivity estimates
- QC CONVMED: the median of the conversion factor estimates
- QC THROMED: the median of the throughput estimates
- QC FWHMPIX COMBINED: median FWHM of cross-dispersion profile in the combined frame (in pixels)
- QC FWHMARC COMBINED: median FWHM of cross-dispersion profile in the combined frame (in arcsec)
- QC FWHMPIX PROFILE: median FWHM of the extraction profile (in pixels)
- QC FWHMARC PROFILE: median FWHM of the extraction profile (in arcsec)
- QC FWHM DIFF: QC FWHMPIX PROFILE - QC FWHMPIX COMBINED

9.12.4 Parameters

- display: Chip for which the results are plotted (default is none)
- hor_size: box size for profile computation (default is 31)
- spec_hsize: Spectrum half size (default is 20)
- closing_hs: Closing kernel half size for BGD cleaning (default is 2)
- degree: degree of the polynomial for the solution (default is 2)
- kappa: kappa for background removal (default is 1.0)
- clean_rate: Minimum rate of bgd pixels per line allowed (default is 0.5)
- rej: Left and right rejections (default is 0,0)
- spec_zone: Spectrum position zone specification (default is -1,-1)
- comb_used: Combined image to be used : NODA / NODB / COMB (default is COMB)
- refine: Flag to refine the offsets (default is TRUE)
- onlyA: Also combine using only nodding A (default is FALSE)
- onlyB: Also combine using only nodding B (default is FALSE)

10 Algorithms

In this section the data reduction procedures applied by the pipeline recipes are described in some detail.

10.1 Model computation

The CRIRES physical model (CPM) and associated algorithms provide an alternative method of wavelength calibration as well as a means of monitoring the instrument behaviour. The model itself is a highly simplified ray trace of the spectrograph optics. Supporting functions allow the iterative evaluation of the ray trace in order to create a wavelength map for the detectors, derive the locus of the spectrum on the detectors and automatically determine the actual instrument configuration from calibration source exposures obtained contemporaneously to science exposures.

10.1.1 Where the CRIRES Physical Model is Used

The principle use of CPM is in calculating the wavelength scale for a given exposure. This occurs automatically (as long as a CPM config file is present in the SOF) in crires_spec_jitter (section 9.8) alongside the calculation of the wavelength scale from a polynomial model. Both scales are present as columns in the output table product, the user may decide which wavelength scale he wants to use. Similarly the crires_model_wlmap iteratively calls the CPM to build up a map of wavelengths on the detector plane for a given config.

The optimal parameter set for CPM (hereafter the CPM "config") is derived from a comprehensive set of Th-Ar HCL exposures taken across the full range of prism and grating values. These are analysed off-line, i.e. the pipeline recipes do not facilitate this task. However, the process uses the CPM (with a previous config that provides at least an approximate fit to the data) firstly to identify spectral lines and then as part of an iterative optimisation algorithm that attempts to minimise the difference between predicted and measured line centroids by continually adjusting the parameter values (using the "simulated annealing" technique). The result is the default CPM config made available to users for use in pipeline recipes as a FITS table. A new default config is derived and made available whenever there is a significant physical change to the instrument (eg. due to a maintenance or upgrade intervention). An example of a CPM config is given in appendix B.

The CRIRES spectral format changes over time, moreover the fit of CPM using the default config is not perfect over all settings (i.e. all prism and grating orientations). Therefore crires_model_refine and crires_model_fix (see 9) exist to allow the user to customise the fit for his data.

10.1.2 The Simplified Ray Trace

The simple optical model implemented in CPM follows closely the prescription of [11] as described in [12]. The system is simplified to a set of focusing elements, dispersive surfaces and detector arrays. This is sufficient to accurately predict the mapping from entrance slit position and wavelength to detector x, y co-ordinates. Only the principle ray is considered and as a result the model is not suitable for, for example, computing beam footprints or PSFs.

The advantage of this simplistic optical model is that the mapping from entrance slit position and wavelength to detector x, y co-ordinates can be described by a manageable number of parameters with physically meaningful

interpretations. A full featured optical model (eg. the CRIRES CodeV optical design) would require a much larger set of parameters. Moreover, CPM is used in the pipeline to perform this mapping iteratively (in the computation of wavelength maps or the optimisation process, see 10.1.4), sometimes during the execution of a recipe 10^9 rays will be traced through the system, this would take much to long with codeV.

In the discussion below we refer to many of the CPM config parameters with the `names` used in the configs. In principle there is nothing to stop users manually editing these parameters, however we do not recommend this.

The principle components of CPM are listed here:

Focusing optics Whilst in practise the collimator and camera of CRIRES are sophisticated, multi-component mirror/lens systems, for our purposes it is sufficient to replace each with a single parameter, the effective focal length, `fcol` and `fdet` for the collimator and camera respectively. `fcol` then allows us to compute the vector for rays entring from anywhere in the slit plane as they approach the prism, while `fdet` determines the scale for the diffracted light on the detector plane.

Prism transformations At the entrance to and exit from the prism the vector undergoes a refraction transformation. At each surface the ray is described by a vector, V, in the frame of the surface. Therefore we can define:

$$\phi = \arctan\left(\frac{V_z}{\sqrt{V_x^2 + V_y^2 + V_z^2}}\right) \tag{1}$$

and:

$$\theta = \arctan\left(\frac{V_x}{V_y}\right) \tag{2}$$

We then apply Snell's law to ϕ:

$$\phi' = \arcsin\left(\frac{sin(\pi/2 - \phi)}{n}\right) \tag{3}$$

where n is the refractive index, and we know that θ is unchanged by the refraction. The refracted vector, V' is then:

$$\begin{aligned} (\text{The wavelength is unchanged}) \qquad V_0' &= V_0 \\ V_x' &= sin(\theta)cos(\pi/2 - \phi') \\ V_y' &= cos(\theta)cos(\pi/2 - \phi') \\ V_z' &= sin(\pi/2 - \phi') \end{aligned} \tag{4}$$

The refractive index, n, is computed from laboratory measurements ([8]) that accurately characterise the ZnSe refractive index as a function of wavelength using the Sellmeier formulation over a range of cryogenic temperatures appropriate to CRIRES operations.

Grating dispersion The incident vector is in the frame of reference of the grating surface. To compute the exit vector we follow [11] and multiply by the Echelle grating matrix:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -m * \sigma & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \tag{5}$$

where m is the spectral order and σ the grating spacing (stored as `sg` in the CPM config) and re-normalise.

Detector array The diffracted ray is projected onto the detector plane as described under *Focusing Optics* above. A bi-dimensional distortion is applied as follows

$$x' = pc_{x,xx}[N].x^2 + pc_{x,x}[N].x + pc_{x,yy}[N].y^2 + pc_{x,y}[N].y + pc_{x,xy}[N].xy \quad (6)$$
$$y' = pc_{y,xx}[N].x^2 + pc_{y,x}[N].x + pc_{y,yy}[N].y^2 + pc_{y,y}[N].y + pc_{y,xy}[N].xy$$

where $pc_{x,xx}[N]$ is the x' coefficient for the x^2 term for the Nth detector chip, $pc_{y,xy}$ is the y' coefficient for the xy term and so on. $pc_{x,xx}[N]$ is stored in the config as `pc_x_xx[N]` etc. In practice the squared terms are not used.

The pixel co-ordinates X, Y on a given detector chip are then given by:

$$X = \frac{S_x}{2} + \frac{(cos(\theta[N])(x' - (x_{chip[N]} + x_{array})) - sin(\theta[N])(y' - (y_{chip[N]} + y_{array})))}{s_{pix}} \quad (7)$$

and:

$$Y = \frac{S_y}{2} + \frac{(cos(\theta[N])(y' - (y_{chip[N]} + y_{array})) - sin(\theta[N])(x' - (x_{chip[N]} + x_{array})))}{s_{pix}} \quad (8)$$

Here $S_{x(/y)}$ is the detector array size in pixels on the $x(/y)$ axis, $\theta[N]$ is the rotation angle of the Nth chip (`chiprot[N]`), $x(/y)_{chip[N]}$ is the relative $x(/y)$ position of the Nth chip (`chipx[N]`/`chipy[N]`), $x(/y)_{array}$ is the absolute $x(/y)$ location of the centre of the detector array (`det_array_x`/`det_array_y`) and s_{pix} is the pixel size (`pix`).

10.1.3 Line Matching

The crires_model_refine recipe (see 9) implements a complex procedure for robust line identification that utilises the CPM.

Determine the instrument setup The prism and grating orientation encoder values along with the intermediate slit width and default order are read from the exposure FITS header.

Calculate exact centroids The CPM is called iteratively for all wavelengths in the line catalogue (only a small fraction will be within the wavelength range of interest) and for the default config and instrument set up. The predicted detector co-ordinates for each spectral feature is stored.

Generate 2D simulated data As above, the CPM is called iteratively for all wavelengths in the line catalogue and for the default config, but this time for adjacent spectral orders (+/-1) as well as the default order. For any wavelengths that fall on the detector array a full Monte-Carlo simulation is performed for photons illuminating the slit, the total number of photons at any wavelength is scaled to the relative intensity given by the line catalogue. In this way a 2D simulated exposure is built up which includes potential order overlap features. (Note that the intermediate slit will prevent most lines from adjacent orders reaching the detector in the CPM just as in the real instrument).

Extract 1D simulated spectrum The spectral range for the given settings is computed and a loop over wavelength values in this range with a step size corresponding to less than one detector pixel is set up. Within the loop, the detector position for a photon passing the (*x* and *y*) centre is computed at the current wavelength. The flux from the pixel in 2D *simulated* data (from the preceding step) is extracted. An interpolation is performed to obtain the flux at the (*x*) pixel centre. This is repeated for 8 pixels above and below and the values extracted are summed. In this way a 1D spectrum is extracted.

Measure centroids in 1D spectrum It is to be expected that location of features in the 1D simulated spectrum correspond to the list of exact centroids. The corresponding measured centroid in the 1D simulated spectrum is computed as:

$$\bar{x} = \frac{\sum_{i=-n}^{n} f_i x_i}{\sum_{i=-n}^{n} f_i} \tag{9}$$

f_i is the flux in pixel i and x_i is the distance of the pixel i from the exact centroid. This is the centroid of all flux within a window of size *n* centered on the exact centroid.

Compare and select for customised list For a well resolved isolated spectral feature, the centroid measured in the simulated 1D spectrum will match that in the list of exact centroids. However, there will be a significant discrepancy if either a nearby neighbour (or an order overlap feature) of non-negligible relative flux contributes flux within this window or the spectral feature in question is too faint to provide an accurate centroid. Therefore features are selected for the customised line list where the discrepancy is <0.1 pixels, once selected they are assigned the exact centroid computed by CPM (*not* the value measured in the simulated data).

As well as enabling the exclusion of actual blends, this approach can also be used to determine whether a calibration feature is unique within a given search box (useful for eliminating false matches in the case that the initial guess calibration has a large uncertainty). For this reason n takes the value passed as the search box size.

Determine global offset The centroids of features extracted from the real data may be uniformly shifted with respect to the centroids predicted with the CPM for the customised line list. We cross-correlate the two lists in order to find the mean offset that provides the highest number of close, unique, matches. This step is repeated for the spectrum from each of the 6 visible calibration fibres.

Compare and select for optimisation list For each centroid predicted by the CPM, the difference to the centroid extracted from the real data is computed after correction for the global offset. If this difference is less than 3.0 pixels then this feature is considered to be reliably matched. The minimum number of matches required is a recipe option. This step is repeated for the spectrum from each of the 6 visible calibration fibres.

10.1.4 Optimisation

Both the crires_model_refine and crires_model_fix recipes use an optimisation algorithm to adjust the CPM parameters to best reproduce the centroids of calibration features in the users specific data.

The user data will only provide a limited number of data points with which to constrain the CPM config so clearly only a subset of the parameter set can be optimised. How many and exactly which parameters are

optimised depends upon how many spectral calibration features have been matched. The following table lists the possibilities:

Number of matches	Open Parameters	limits
1	Grating angle coeff	+/-0.1°
2	Grating angle coeff	+/-0.1°
	Prism angle coeff	+/-Equiv 100enc
3	Grating angle coeff	+/-Equiv 100enc
	Camera focal length (`fk`)	+/-10mm
	Grating constant (`sg`)	+/-1.0mm$^-$1
4	Grating angle coeff	+/-Equiv 100enc
	Camera focal length (`fk`)	+/-10mm
	Grating constant (`sg`)	+/-1.0mm$^-$1
	Prism angle coeff	+/-Equiv 100enc
>5	Grating angle coeff	+/-Equiv 100enc
	Camera focal length (`fk`)	+/-10mm
	Grating constant (`sg`)	+/-1.0mm$^-$1
	Prism angle coeff	+/-Equiv 100enc
	Detector chip x centres (`chipx[`N`]`)	$\pm$0.5mm

Note that the open parameters are *not* the prism and grating angles, but instead the zeroth order coefficients. This is because when the CPM is evaluated in recipes such as crires_spec_jitter or crires_model_wlmap, the prism and grating angles are computed from the encoder values in the science file header using the angle-encoder relations defined by a set of coefficients that are also contained in the CPM config. The chip x centres are only opened for those chips for which there are at least two identified calibration features (from a single fibre).

Once the open parameters and limits have been determined, the simulated annealing technique (See [9] for further details.) is employed to iteratively call the CPM while continually adjusting the open parameter values so that the discrepancy between measured and predicted calibration line centroids is minimised.

10.1.5 Wavelength Maps and Loci of Spectra

Recipes that compute wavelength maps, the locus of a spectrum for a given entrance slit position or the wavelength solution along that locus use the assumptions and logic outlined below.

Wavelength range The wavelength range considered is from the blaze wavelength of the order immediately above the given order (λ_{low}) and that of the order immediately below (λ_{high}):

$$\lambda_{low} = \frac{\left(sin\left(\mu_g - \arctan\left(\frac{d}{f}\right)\right) + sin\left(\mu_g + arctan\left(\frac{d}{f}\right)\right)\right)}{s_g * (m+1)} \tag{10}$$

$$\lambda_{high} = \frac{\left(sin\left(\mu_g - \arctan\left(\frac{d}{f}\right)\right) + sin\left(\mu_g + arctan\left(\frac{d}{f}\right)\right)\right)}{s_g * (m-1)}$$

where μ_g is the grating angle in degrees, d is the grating deflection distance in mm, f is the focal length of the camera optics in mm, s_g the grating spacing in mm and m the spectral order.

Wavelengths assigned to pixels in wavelength maps The entrance slit cross-dispersion position is set to loop over a range of positions covering the entire slit length (the dispersion position is set to the centre of the slit width) and λ iterates between the ranges defined above. In both cases the increment is set to correspond to less than one pixel on the detector array. As the centre of each pixel is passed in the dispersion direction the wavelength that would have fallen at the centre is interpolated. There is no interpolation in the cross-dispersion direction.

Determination of physical entrance slit position The CPM requires the entrance slit position of the spectrum source along the length of the slit in mm (s_x). The position of the target in arc seconds along the slit ought to be available in the science FITS file header, in which case it can be simply converted. However, it is possible that this information is unavailable (the spectrum of interest may not be that of the target). In this case we take the y (direction perpendicular to dispersion) pixel coordinate of the spectrum trace at the x axis (dispersion direction) centre of the 3rd detector chip. From this the appropriate value of s_x (for the given instrument configuration) can be determined. Moreover, in this way we do not require that the main pipeline translates the angular position on the slit as computed by the instrument software to the distance in mm as required by the CPM. In order to do this a Newton-Raphson interpolation is used to iteratively call CPM and find the value of s_x corresponding to the y co-ordinate that was passed.

Wavelength solution along a locus This is similar to the generation of a wavelength map except that there is no loop over the entrance slit position, since it can be fixed at the value determined as described above. Wavelengths along the locus can then be obtained directly from the wavelength map.

10.2 General Algorithms

10.2.1 Wavelength calibration

Inputs The wavelength calibration is either computed using some calibration data obtained with Thorium/Argon lamps or with a N2O gas cell, or using directly the sky emission lines obtained from the observation. These lines are then compared to a theoretical signal extracted from catalogs of lines to obtain the wavelength calibration. The advantage of using the sky lines is that the conditions are exactly the same as for the observation. This assures a better accuracy.
For each setting (identified from the header), the configuration table gives a first guess of the wavelengths of the first and the last pixels (first degree polynomial).

The list of inputs to find the wavelength-pixels relation is the following:

- The extracted spectrum in pixels (Figure 10.2.1)
- The lines catalog (Figure 10.2.1)
- The first guess estimation of the first and last pixels wavelengths (WL_min and WL_max) obtained from the header

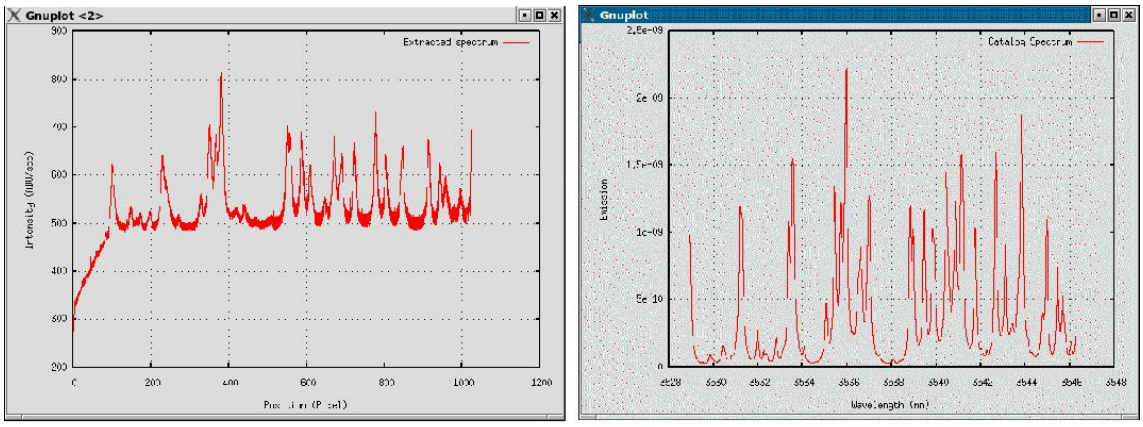

Figure 10.2.1: *Wavelength calibration inputs.*

- The requested polynomial degree (degree)
- The wavelength error defining the precision of the first guess
- the number of samples used to define the candidate polynomials (nsamples)

Algorithm As shown on Figure 10.2.2, ndegree+1 points (A0, A1, A2 in our example) are defined in the plane (wavelength,pixel).

These points are equally spaced along the pixel direction. The corresponding wavelengths are typically obtained by a linear solution based on the information found in the header (INS.WLEN.STRTi, INS.WLEN.ENDi, where i = 1,2,3,4). Then $nsamples$ (7 in the example illustrated by Figure 10.2.2) are defined around these initial guesses in the wavelength direction.

The $nsamples$ positions are regularly spaced not further than the specified wavelength error (2.0 nm in our example) from the initial guess points A0, A1, A2.

These $nsamples$ points define a number ($nsamples^{ndegree+1} = 7^3 = 343$) of candidate polynomials as shown Figure 10.2.2.

For each of these candidate polynomials, the extracted spectrum is converted from pixels to wavelength and directly compared to the catalog. A likelihood factor is computed and associated to the candidate polynomial (Figure 10.2.3 gives an example of computed likelihood factors).

The maximum defines the polynomial that is the closest to the real solution.

Figure 10.2.4 shows how the initial guess has been corrected to align the extracted spectrum on the catalog.

On the computation efficiency point of view, specifying a big number of samples together with a high degree for the polynomial explodes the number of candidate polynomials to evaluate. On the other hand if the first guess is not very precise, it is necessary to use a large wavelength error. Therefore, lots of samples are needed to obtain a best polynomial close to the real solution.

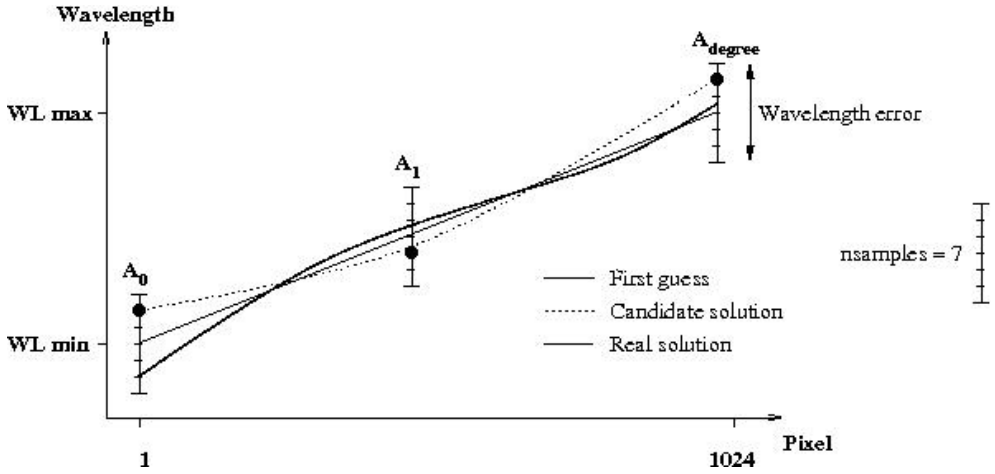

Figure 10.2.2: *Wavelength calibration algorithm.*

For these reasons, we proceed iteratively. A first pass is used with a large wavelength error, a big number of samples (100), but a first degree polynomial to refine the linear first guess (10000 evaluations).

The result is used as first guess in the second pass, where the number of samples is divided by 10 and the requested degree is 2 (1000 evaluations).

Again, the result is used as first guess in the third pass where the wavelength error is divided by 10 but the degree of the requested polynomial (2) and the number of samples (10) remain the same.

This way, we iteratively converge to the searched solution.

10.2.2 Images combination

There are two kind of supported combinations.

The first one is a simple jittered observation where the images are shifted and stacked together.

The second one is more challenging. The images are nodded between two main positions A and B. Depending on their offsets values, the frames are classified as 'A' or 'B'. They are then associated together as 'A and B' pairs and subtracted (A-B and B-A) to obtain 2 so called nodded images.

These nodded images are then shifted and stacked together to create the combined image. Optionally, a combined image can be created out of only the 'A' nodded images (and the same with only the 'B' nodded images).

Figure 10.2.5 describes the re-combination of nodded images.

10.2.3 Spectrum extraction

The spectrum extraction is computed with two different methods in parallel: a simple *rectangular* extraction and an *optimal* extraction that is more adapted to low signal to noise images with very faint sources.

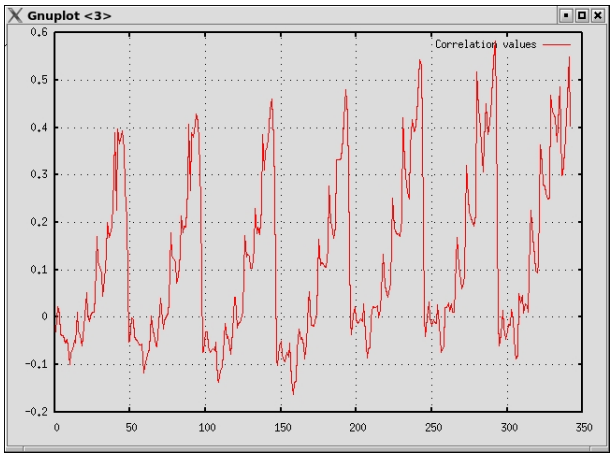

Figure 10.2.3: *Likelihood factors of the candidates* ($nsamples = 7$ *and* $degree = 2$*).*

The two methods have many similarities and are described in parallel in the following paragraphs. The $_{rect}$ and $_{opt}$ are used to distinguish them.

1. Background Noise

   The background noise is compute for each column of the image. For each column, an iterative sigma clipping algorithm is applied to identify which pixels belong or not to the background. At each iteration, statistics are computed on the column, a threshold is computed from those statistics, and the pixels that are too far from the average value are rejected.

   After 5 iterations, the remaining pixels are considered being part of the background.

   Figure 10.2.6 shows the background map obtained with this method.

   A morphological closing is then applied with a kernel size that can be specified by the user, followed by an optional cleaning of the lines that contain too few background pixels.

   For each column c, the noise of the background $bgd\_noise(c)$ is the standard deviation of the pixels identified as background.

2. Spectrum Profile

   An initial simple profile ($prof\_init$) is first computed by applying a median filter to the input image with a kernel of around 30 pixels along the dispersion direction. This filter enhance the horizontal spectrum and makes it easy to detect its position.

   Knowing the spectrum position ($spec\_pos$) and width ($spec\_width$), the *rectangular* extraction profile ($prof\_rect$) can be created as it is simply a rectangular mask covering the spectrum.

   The *optimal* extraction profile ($prof\_opt$) at column c and line l is computed with:

$$prof\_opt(c,l) = \frac{prof\_init(c,l)}{\frac{prof\_init(c,l)}{conv*exptime} + size * bgd\_noise(c)^2} \tag{11}$$

   where:

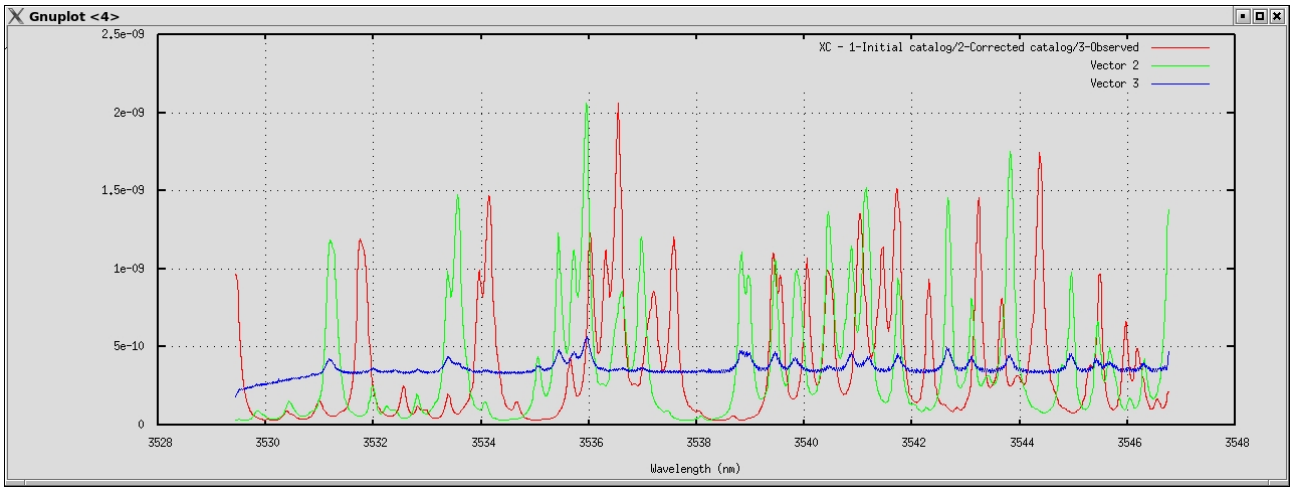

Figure 10.2.4: *Overplot of the catalog and the spectrum.*

- $conv$ is the gain of the detector (respectively $7.737 * 0.9$, $7.664 * 0.9$, $7.689 * 0.9$, $8.077 * 0.9$ for chip 1, 2, 3, 4)
- $exptime$ is DIT x NDIT x NEXP
- bgd_noise is the background noise
- $size$ is the size of the kernel used for the horizontal median filter (31)
- $prof\_init$ is the profile computed at the second step

Eventually, for both $prof\_opt$ and $prof\_rect$, the columns are divided by $w\_opt(c)$ and $w\_rect(c)$, the sum of their pixels.

3. Error

   The *rectangular* and *optimal* extraction errors $error\_rect(c)$ and $error\_opt(c)$ are given by:

$$error\_rect(c) = \sqrt{bgd\_noise(c)^2 * (\sum_l prof\_rect(c,l)^2) + \sum_l (prof\_rect(c,l)^2 * \frac{input(c,l)}{conv * exptime}) * spec\_wi} \quad (12)$$

$$error\_opt(c) = \frac{\sqrt{bgd\_noise(c)^2 * (\sum_l prof\_opt(c,l)^2) + \sum_l (prof\_opt(c,l)^2 * \frac{input(c,l)}{conv * exptime})}}{correction(c)} \quad (13)$$

   where:

   - bgd_noise is the background noise
   - $prof\_rect$ and $prof\_opt$ are the rectangular and optimal profiles
   - $conv$ is the gain of the detector (respectively $7.737 * 0.9$, $7.664 * 0.9$, $7.689 * 0.9$, $8.077 * 0.9$ for chip 1, 2, 3, 4)

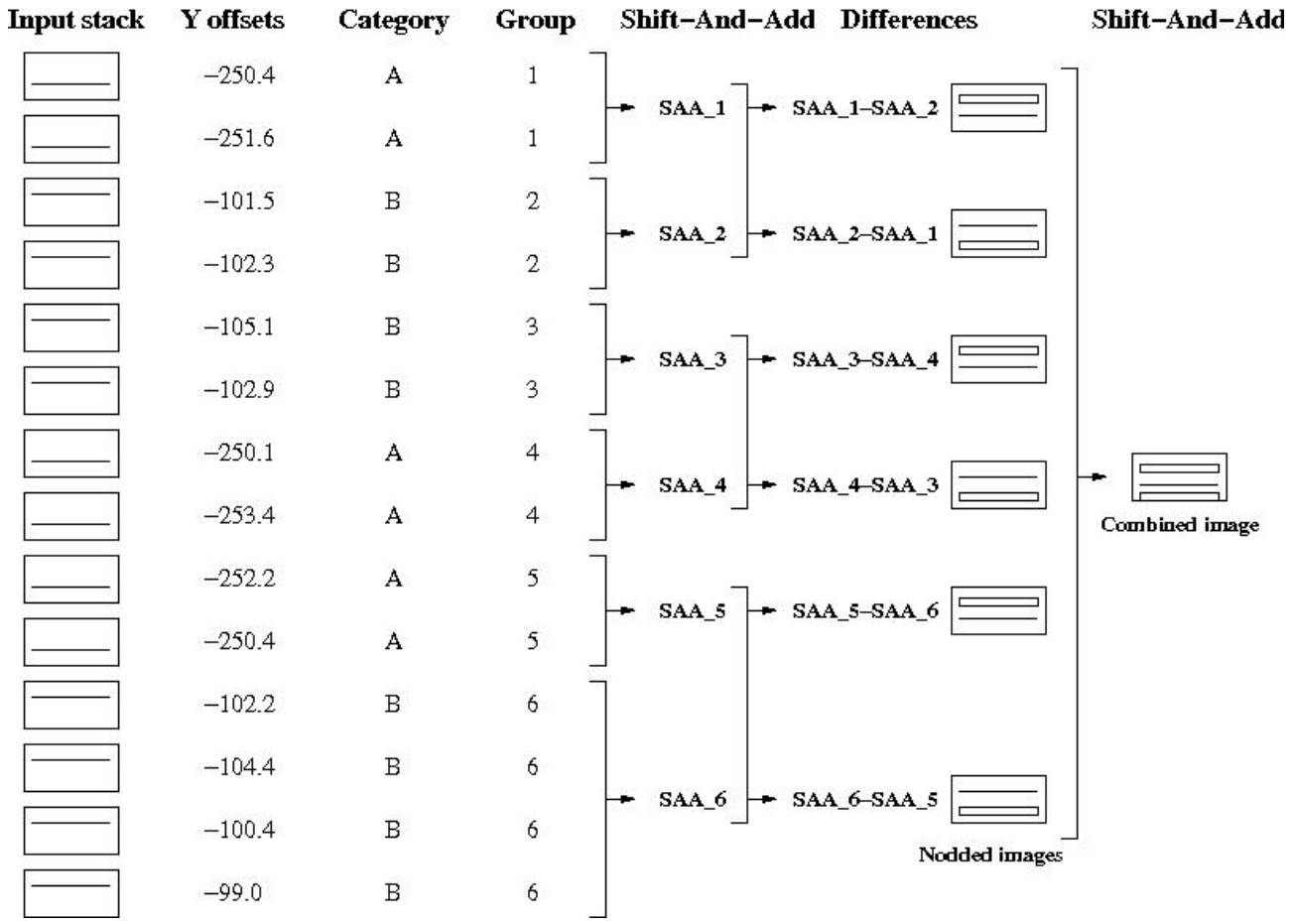

Figure 10.2.5: *Nodded images combination.*

- $exptime$ is DIT x NDIT x NEXP
- $input$ is the input image value
- $correction(c)$ is given by:

$$correction(c) = \sum_{l} prof\_opt(c,l) * prof\_init(c,l) \tag{14}$$

4. Spectrum Intensity

   Finally, the spectrum can be extracted using the 2 different profile with the following formulas:

$$spec\_rect(c) = (\sum_{l} prof\_rect(c,l) * input(c,l)) * spec\_width \tag{15}$$

$$spec\_opt(c) = \frac{\sum_l prof\_opt(c,l) * input(c,l)}{correction(c)} \tag{16}$$

   where:

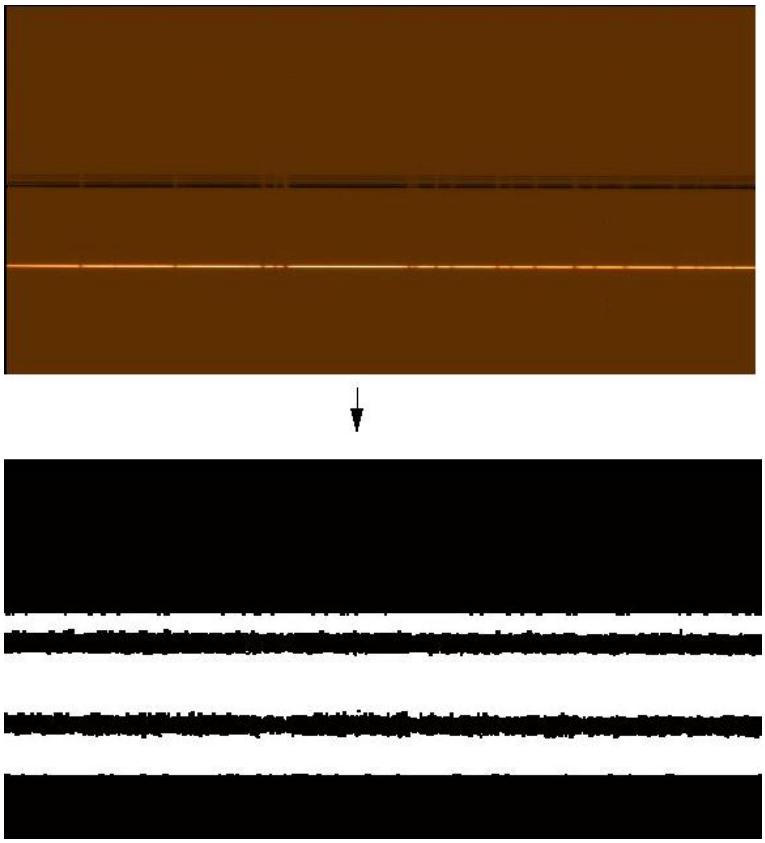

Figure 10.2.6: *Background Identification.*

- $prof\_rect$ and $prof\_opt$ are the rectangular and optimal profiles
- $input$ is the input image value
- $correction(c)$ is given by 14

Figure 10.2.7 shows some plots of the different quantities computed is this process on one example.

10.2.4 Conversion factor, Sensitivity, Throughput

The conversion factor is expressed in ADU/sec/Jy.

As well as for the spectrum extraction, two conversion factors are computed:

$$conversion\_rect(\lambda) = \frac{spec\_rect(\lambda)}{phot\_flux(\lambda)} \tag{17}$$

$$conversion\_opt(\lambda) = \frac{spec\_opt(\lambda)}{phot\_flux(\lambda)} \tag{18}$$

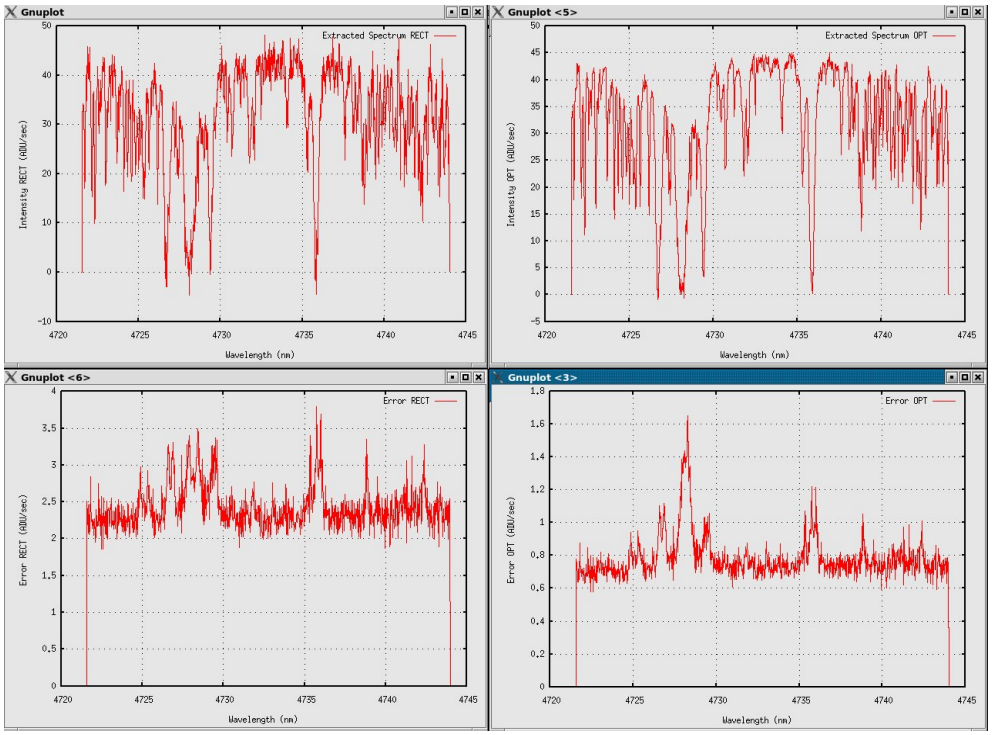

Figure 10.2.7: *Rectangular and Optimal extractions.*

where:

- $spec\_rect$ and $spec\_opt$ are the rectangular and optimal extracted spectra
- $phot\_flux$ is the photospheric flux of the standard star in Jy

The throughput is computed from the rectangular conversion factor and expressed in %:

$$throughput(\lambda) = \frac{conversion\_rect(\lambda) * conv}{\frac{10^{-32} * S * \delta\lambda * 10^6}{\lambda * h}} \tag{19}$$

where:

- $conv$ is the gain of the detector (respectively $7.737 * 0.9$, $7.664 * 0.9$, $7.689 * 0.9$, $8.077 * 0.9$ for chip 1, 2, 3, 4) in electrons/ADU
- S is the suface of the primary miror in square meters
- $h = 6.626.10^{-34} J.s$ is the Planck constant
- $\delta\lambda$ is the dispersion (spectral coverage of one pixel) in meters per pixels

The sensitivity in Jy/10 sigma/hour is computed from the optimal conversion factor:

$$sensitivity(\lambda) = \frac{10 * error\_opt(\lambda) * \sqrt{\frac{exptime}{3600}}}{conversion\_opt(\lambda)} \quad (20)$$

where:

- $error\_opt$ is the optimal extraction error
- $exptime$ is DIT x NDIT x NEXP
- $conversion\_opt$ is the optimal conversion factor

10.3 Recipes Algorithms

10.3.1 crires_spec_dark

Dark frames are exposures without detector illumination. The dark current of the CRIRES detector is small, so the dominant feature in these frames is the detector bias.

The darks are reduced with the `crires_spec_dark` recipe.

For each chip, the recipe produces one master dark that is nothing more than an average of the input files divided by the DIT. The Read-Out Noise (RON) is also measured and written as QC parameter. For each chip, the first image is subtracted from the second, and the second from the third. The following measurement is applied to those 2 differences to produce RON1 and RON2:

- Generate 100 13x13 windows on the input pixel surface. These windows are optimally distributed using a Poisson distribution to make sure they sample the whole area with as little overlap as possible.
- Compute the pixel standard deviation in each window.
- The readout noise is the median of all these measured standard deviations multiplied by $\sqrt{\frac{NDIT}{2}}$.

The results are written as QC parameters in the different product extensions.

Additional QC parameters are the average and the standard deviation of the median values of the input images (QC DARKMED and QC DARKSTDEV).

10.3.2 crires_win_dark

This recipe is used to reduce data acquired using the read-out mode *FowlerNsampGRstWin*. The same algorithm as *crires_spec_dark* is used on chips 2 and 3.

10.3.3 crires_spec_flat

A dark frame can be passed to the recipe to initially correct for the dark. The detector non-linearity should also be removed by passing the polynomial coefficient frames describing this non-linearity.

For each chip, the list of images is loaded in memory, each individual image is divided by the median value of its central part. If the vignetting is different than the one expected, the *starty* and *stopy* parameter can be use to specify the zone use to compute the median used for normalisation.

The image list is then collapsed to one image. The collapsing method is specified by the option *coll_meth*. It can be a simple average (use "avg"), a median (use "med") or a more complex sigma clipping algorithm (use "sig"). The sigma-clipping algorithm is recommended and used by default.

This resulting image is the flatfield for the given chip.

The 4 chips flatfields are stored in one 4 extensions file which is the CRIRES master flat.

Additionally, a thresholding is computed on this flat to identify the bad pixels. A 4 extensions bad pixel map is produced.

The QC parameters are produced for each chips and stored in the corresponding chip extension of the products. They contain the standard deviation of the final flat (QC FLAT MASTER RMS), the average and standard deviation of the median values of the input frames (QC FLAT MEAN and QC FLAT STDEV) and the number of bad pixels (QC NBBAD).

10.3.4 crires_win_flat

This recipe is used to reduce data acquired using the read-out mode *FowlerNsampGRstWin*. The same algorithm as *crires_spec_dark* is used on chips 2 and 3.

10.3.5 detmon_ir_lg

Please refer to [1] to have a detailed description of this generic and instrument independant recipe.

10.3.6 crires_spec_wavecal

This recipe performs the wavelength calibration of the CRIRES detectors.

The input raw frames can be corrected from the dark, flat, bad pixels and detector non-linearity if the relevant calibration files are provided to the recipe.

The recipe can accept 3 different types of raw frames as input:

1. Sky observation

   In this case, the sky lines are used to derive the wavelength calibration. The OH lines catalog is used if the central wavelength is below 1.8 microns, otherwise the HITRAN catalog is used.

2. Lamp calibration

   If the calibration is done with the Thorium/Argon lamp, the spectrum is extracted at the various pinhole positions along the slit. The catalog used is the one containing the Thorium/Argon lines.

3. Gas cell

   A N2O gas cell can be brought in front of the halogen lamp to generate N2O absorption lines. The N2O catalog is used.

In the 3 cases, the wavelength calibration is computed at a number of positions along the slit. These positions are hardcoded to the pinhole positions of the Th/Ar calibration. The user can specify a number of positions on the command line with e.g. y_pos_c2="50,150,234,350,400,500" for detector number 2).

For each wavelength calibration, the spectrum is extracted by collapsing the image rows around the specified position.

The problem is now reduced to comparing an extracted observed signal that represents lines at given pixel positions with the catalog containing the same lines at given wavelengths.

This problem is solved with the iterative cross-correlation method described in details in section 10.2.1.

Figure 10.3.1 shows an example of the produced solution, and Figure 10.3.2 illustrate a result by surimposing the extracted lines spectrum and the catalog signal in the case of a sky lines example.

10.3.7 crires_model_refine

This recipe applies a detection of the Thorium/Argon emission lines from a calibration lamp exposures, and uses a cross-check of the detected lines with the lines catalog to perform an optimisation of various model parameters.

After those parameters are optimised, they are saved in a new model configuration file that can be used to calibrate subsequent science observations.

10.3.8 crires_spec_jitter

This recipe is the main observation data reduction tool. The algorithms described in the following are applied to the different chips separately, all the intermediate and final products have the same format as the raw CRIRES data: 4 extensions for the different chips.

The following successive steps are performed:

1. Calibrations

   If provided as input calibration files, the following corrections are applied to the input data: flat, dark, bad pixels correction and detector linearity correction.

2. Data combination

```
#
# file          crires_spec_wavecal_tab.fits
# extensions    4
# ---------------------------------------------
# XTENSION      1
# Number of columns 7
#
   Coeff_a|   Coeff_b|   Coeff_c|   Coeff_d|   Coeff_e|Y_Position|XCorrelation
   1073.84|0.00548974|         0|         0|         0|        84|          -1
   1073.84|0.00548974|         0|         0|         0|       162|          -1
   1073.84|0.00548974|         0|         0|         0|       242|          -1
   1073.84|0.00548974|         0|         0|         0|       322|          -1
   1073.84|0.00548974|         0|         0|         0|       400|          -1
   1073.84|0.00548974|         0|         0|         0|       480|          -1
# ---------------------------------------------
# XTENSION      2
# Number of columns 7
#
   Coeff_a|   Coeff_b|   Coeff_c|   Coeff_d|   Coeff_e|Y_Position|XCorrelation
   1081.01|0.00526491|         0|         0|         0|        80|          -1
   1081.03|0.00522135|1.74972e-07|         0|         0|       160|     0.57307
   1081.03|0.00531624|-6.50722e-08|         0|         0|       240|     0.28812
   1081.03|0.00530908|-5.7842e-08|         0|         0|       320|    0.433405
   1081.02|0.00531809|-6.45902e-08|         0|         0|       398|    0.631744
   1081.01|0.00526491|         0|         0|         0|       476|          -1
# ---------------------------------------------
# XTENSION      3
# Number of columns 7
#
   Coeff_a|   Coeff_b|   Coeff_c|   Coeff_d|   Coeff_e|Y_Position|XCorrelation
    1087.8|0.00508404|-3.59359e-08|         0|         0|        66|    0.972271
   1087.82|0.00507507|-9.67506e-09|         0|         0|       146|    0.991676
   1087.82|0.00508981|-3.96217e-08|         0|         0|       226|    0.991876
   1087.83|0.00508604|-2.81037e-08|         0|         0|       306|    0.992499
   1087.83|0.00509335|-4.14645e-08|         0|         0|       384|    0.991144
   1087.82|0.00509866|-5.02182e-08|         0|         0|       462|    0.982228
# ---------------------------------------------
# XTENSION      4
# Number of columns 7
#
   Coeff_a|   Coeff_b|   Coeff_c|   Coeff_d|   Coeff_e|Y_Position|XCorrelation
   1094.29|0.00479374|         0|         0|         0|        60|          -1
    1094.3|0.00482925|-3.51104e-09|         0|         0|       139|    0.348436
   1094.31|0.00479352|-2.36995e-08|         0|         0|       217|    0.279748
   1094.29|0.00479374|         0|         0|         0|       297|          -1
   1094.31|0.00480475|2.01885e-08|         0|         0|       376|    0.303523
   1094.29|0.00479374|         0|         0|         0|       455|          -1
```

Figure 10.3.1: *Example of a wavelength calibration product.*

Depending on the observation mode, the data are combined together in one combined image. The combination is slightly different if there is jittering or not and if there is nodding or not.

The offsets between the different frames are read from the headers, and refined using bright features in the image.

Additionally, a contribution map provides the number of frames used for each pixel of the combined image.

The combination method is described in section 10.2.5.

3. Spectrum extraction

   From the combined image, the brightest spectrum is extracted using two different methods: a rectangular extraction method, and a more complex one adapted for faint spectra: the optimal extraction.

   These extraction methods are described in section 10.2.3.

4. Wavelength calibration using cross-correlation

   The user can provide as input a wavelength solution (computed e.g. by crires_spec_wavecal). In this case, the wavelength computation is skipped, and the provided solution is used.

   Otherwise, the sky lines are extracted from the first input frame, and correlated with the provided OH lines or HITRAN catalog depending if the central wavelength is lower or higher than 1.8 microns.

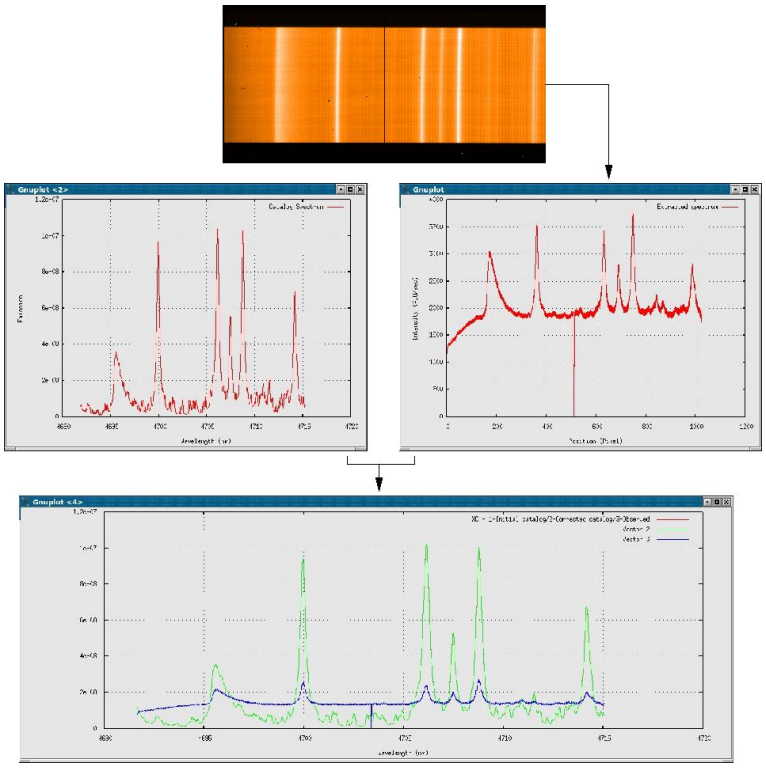

Figure 10.3.2: *Wavelength calibration.*

See section 10.2.1 for more details.

5. Wavelength computation using the model

   Provided that the user gives as input the configuration file for the model, this one is run to compute the wavelengths for every pixel on the detector. A wavelength map is produced, as well as a table with the wavelength for each pixels along the extracted spectrum.

6. Conversion, sensitivity and throughput computations for standard stars.

   In case of a standard star observation, and if the photospheric flux catalog has been provided, the conversion curve, the sensitivity and the throughput are computed following the algorithm described in section 10.2.4.

Figure 10.3.3 shows a sketch of the algorithm applied by the jitter.

10.3.9 crires_win_jitter

This recipe is used to reduce data acquired using the read-out mode *FowlerNsampGRstWin*. The same algorithm as *crires_spec_jitter* is used on chips 2 and 3 with two limitations.

First, the recipe does not compute the wavelength, it expects the wavelength solution to be provided as input.

Second, the jitter image combination is processed on the 2 stripes separately, and the 2 stripes of the 2 combined images are put back together to generate the combined image, the contribution map, and all the image products.

10.3.10 crires_spec_astro

This recipe first sorts out the input raw frames according to the derotator angle. For each of the groups, it applies the same data reduction algorithm as for the crires_spec_jitter recipe (see 10.3.8).

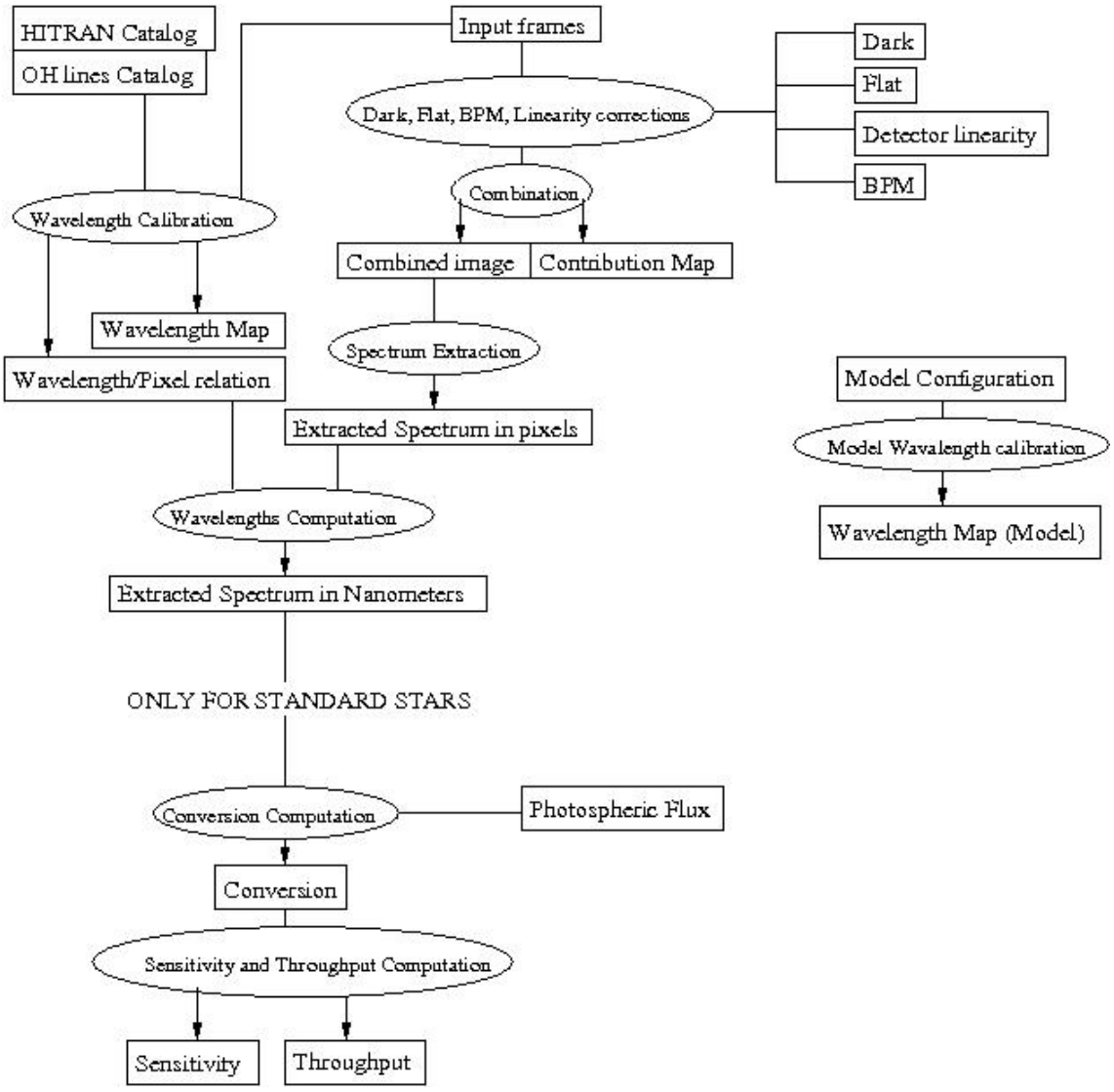

Figure 10.3.3: *Description of the algorithm applies by crires_spec_jitter.*

A Installation

This chapter gives generic instructions on how to obtain, build and install the CRIRES pipeline.

The supported platforms are listed in Section A.1. It is recommended reading through Section A.2.2 before starting the installation.

A bundled version of the CRIRES pipeline with all the required tools and an installer script is available from `http://www.eso.org/pipelines/`, for users who are not familiar with the installation of software packages.

A.1 Supported platforms

The utilisation of the GNU build tools should allow to build and install the CRIRES pipeline on a variety of UNIX platforms, but it has only been verified on the VLT target platforms:

- Linux (glibc 2.1 or later),
- Sun Solaris 2.8 or later,

using the GNU C compiler (version 3.2 or newer).

A.2 Building the CRIRES pipeline

This section shows how to obtain, build and install the CRIRES pipeline from the official source distribution.

A.2.1 Requirements

To compile and install the CRIRES pipeline one needs:

- the GNU C compiler (version 3.2 or later),
- the GNU `gzip` data compression program,
- a version of the `tar` file-archiving program, and,
- the GNU `make` utility.

An installation of the Common Pipeline library (CPL) must also be available on the system. The CPL distribution can be obtained from `http://www.eso.org/cpl`.

Please note that CPL itself depends on an existing cfitsio installation.

In order to run the CRIRES pipeline recipes a front-end application is also required. Currently there are two such applications available, a command-line tool called *EsoRex* and the Java based data file organizer, *Gasgano*,

which provides an intuitive graphical user interface (see Section 4.2, page 17). At least one of them must be installed. The *EsoRex* and *Gasgano* packages are available at `http://www.eso.org/cpl/esorex.html` and `http://www.eso.org/gasgano`, respectively.

For installation instructions of any of the additional packages mentioned before please refer to the documentation of these packages.

A.2.2 Compiling and installing the CRIRES pipeline

The CRIRES pipeline distribution kit 2.3.13contains:

crire-pipeline-manual-1.14.pdf	The CRIRES pipeline manual
install_pipeline	Install script
cfitsio3490.tar.gz	CFITSIO 3490
cpl-7.1.4.tar.gz	CPL 7.1.4
esorex-3.13.4.tar.gz	esorex 3.13.4
gasgano-2.4.8.tar.gz	GASGANO 2.4.8
crire-2.3.13.tar.gz	CRIRES 2.3.13
crire-calib-2.3.13.tar.gz	CRIRES calibration files 2.3.13

Here is a description of the installation procedure:

1. Change directory to where you want to retrieve the CRIRES pipeline recipes package. It can be any directory of your choice but not:

   ```
   $HOME/gasgano
   $HOME/.esorex
   ```

2. Download from the ESO ftp server, `http://www.eso.org/pipelines/`, the latest release of the CRIRES pipeline distribution.

3. Verify the checksum value of the tar file with the cksum command.

4. Unpack using the following command:

   gunzip crire-kit-2.3.13.tar.gz

   tar -xvf crire-kit-2.3.13.tar

5. Install: after moving to the top installation directory,

   cd crire-kit-2.3.13

   it is possible to perform a simple installation using the available installer script (*recommended*):

   ```
   ./install_pipeline
   ```

(beware: the execution may take a few minutes on Linux and several minutes on SunOS).

By default the script will install the CRIRES recipes, *Gasgano*, *EsoRex*, all the necessary libraries, and the static calibration tables, into a directory tree rooted at `$HOME`. A different path may be specified as soon as the script is run.

The only exception to all this is the *Gasgano* tool, that will always be installed under the directory `$HOME/gasgano`. Note that the installer will move an existing `$HOME/gasgano` directory to `$HOME/gasgano` before the new *Gasgano* version is installed.

Important: the installation script would ensure that any existing *Gasgano* and *EsoRex* setup would be inherited into the newly installed configuration files (avoiding in this way any conflict with other installed instrument pipelines).

Alternatively, it is possible to perform a manual installation (*experienced users only*).

B Model Configuration File

Any use of the physical model (whether directly through the crires_model_ recipes or as a step in the pipeline using one of the crires_spec_ recipes) requires a model configuration FITS file to be tagged with CALPRO_MODEL_CONFIG in the input set of frames. This file has the following five columns:

- Parameter Value
- Lower limit
- Upper limit
- Flag
- Parameter Name

The values in columns 2,3 and 4 are only relevant to the configuration optimisation process which is beyond the scope of this document. Their values are ignored by the functions and recipes discussed above. Below is an example model configuration file containing all the necessary parameters required by the above functions and recipes (the values in column 1 are just examples, they happen to be the values in use at the time of writing).

```
#
# file           model_conf.fits
# extensions     1
# ----------------------------------------
# XTENSION       1
# Number of columns 5
#
Best_Guess| Low_Limit|High_Limit|Compute_Flag|Parameter_Name
        72|        72|        72|           0|   temper
   3.49563|   3.49563|   3.49563|           0|   es_x
         1|         1|         1|           0|   es_xf
         0|         0|         0|           0|   es_x_off
         0|         0|         0|           0|   es_y
         0|         0|         0|           0|   mues
         0|         0|         0|           0|   nues
  0.245794|  0.245794|  0.245794|           0|   taues
      7.65|      7.65|      7.65|           0|   es_s
     0.062|     0.062|     0.062|           0|   es_w
   742.068|   742.068|   742.068|           0|   fcp
   37.5069|   37.5069|   37.5069|           0|   mup1
 -0.308137| -0.308137| -0.308137|           0|   axis_mup
   37.9718|   37.9718|   37.9718|           0|   penc_co1
3.0048e-05|3.0048e-05|3.0048e-05|           0|   penc_co2
1.14858e-11|1.14858e-11|1.14858e-11|           0|   penc_co3
  0.318643|  0.318643|  0.318643|           0|   nup1
```

```
         0|          0|          0|           0|   taup1
         0|          0|          0|           0|   axis_nup
         0|          0|          0|           0|   axis_taup
  -15.0625|  -15.0625|  -15.0625|           0|   mup2
         0|          0|          0|           0|   nup2
         0|          0|          0|           0|   taup2
         0|          0|          0|           0|   rind_delta
         0|          0|          0|           0|   muis
         0|          0|          0|           0|   nuis
 -0.178371| -0.178371| -0.178371|           0|   tauis
         1|          1|          1|           0|   flipisx
         1|          1|          1|           0|   flipisy
 -0.069293| -0.069293| -0.069293|           0|   fis
  -1.27276|  -1.27276|  -1.27276|           0|   slitx
    52.438|    52.438|    52.438|           0|   slity
        90|         90|         90|           0|   slitrot
         0|          0|          0|           0|   slitw
        40|         40|         40|           0|   slith
         1|          1|          1|           0|   slit_on
   62.4582|   62.4582|   62.4582|           0|   mug
 -0.162869| -0.162869| -0.162869|           0|   axis_mug
    63.343|    63.343|    63.343|           0|   genc_co1
-3.01773e-05|-3.01773e-05|-3.01773e-05|           0|   genc_co2
-3.23983e-13|-3.23983e-13|-3.23983e-13|           0|   genc_co3
0.00243969|0.00243969|0.00243969|           0|   axis_nug
         0|          0|          0|           0|   axis_taug
 0.0706646| 0.0706646| 0.0706646|           0|   nug
0.00760184|0.00760184|0.00760184|           0|   taug
   31.7284|   31.7284|   31.7284|           0|   sg
   103.479|   103.479|   103.479|           0|   cam_dis
   1527.32|   1527.32|   1527.32|           0|   fk
         0|          0|          0|           0|   mud
         0|          0|          0|           0|   nud
        90|         90|         90|           0|   taud
  0.027269|  0.027269|  0.027269|           0|   pix
  -53.6923|  -53.6923|  -53.6923|           0|   chipx[0]
  -2.18493|  -2.18493|  -2.18493|           0|   chipy[0]
 -0.249051| -0.249051| -0.249051|           0|   chiprot[0]
  -17.8265|  -17.8265|  -17.8265|           0|   chipx[1]
  -2.16249|  -2.16249|  -2.16249|           0|   chipy[1]
 -0.228227| -0.228227| -0.228227|           0|   chiprot[1]
   17.6949|   17.6949|   17.6949|           0|   chipx[2]
  -1.83376|  -1.83376|  -1.83376|           0|   chipy[2]
  0.055346|  0.055346|  0.055346|           0|   chiprot[2]
   53.0034|   53.0034|   53.0034|           0|   chipx[3]
```

```
  -1.64966|  -1.64966|  -1.64966|           0|   chipy[3]
 -0.476486| -0.476486| -0.476486|           0|   chiprot[3]
         0|         0|         0|           0|   pc_x_xx[0]
  0.998427|  0.998427|  0.998427|           0|   pc_x_x1[0]
         0|         0|         0|           0|   pc_x_yy[0]
0.00791644|0.00791644|0.00791644|           0|   pc_x_y1[0]
         0|         0|         0|           0|   pc_x_xy[0]
         0|         0|         0|           0|   pc_y_xx[0]
0.00014341|0.00014341|0.00014341|           0|   pc_y_x1[0]
         0|         0|         0|           0|   pc_y_yy[0]
  0.996284|  0.996284|  0.996284|           0|   pc_y_y1[0]
         0|         0|         0|           0|   pc_y_xy[0]
         0|         0|         0|           0|   pc_x_xx[1]
  0.998564|  0.998564|  0.998564|           0|   pc_x_x1[1]
         0|         0|         0|           0|   pc_x_yy[1]
0.00307149|0.00307149|0.00307149|           0|   pc_x_y1[1]
         0|         0|         0|           0|   pc_x_xy[1]
         0|         0|         0|           0|   pc_y_xx[1]
0.000539665|0.000539665|0.000539665|           0|   pc_y_x1[1]
         0|         0|         0|           0|   pc_y_yy[1]
  0.997466|  0.997466|  0.997466|           0|   pc_y_y1[1]
         0|         0|         0|           0|   pc_y_xy[1]
         1|         1|         1|           0|   pc_x_xx[2]
   1.00096|   1.00096|   1.00096|           0|   pc_x_x1[2]
0.00278721|0.00278721|0.00278721|           0|   pc_x_yy[2]
0.00055155|0.00055155|0.00055155|           0|   pc_x_y1[2]
         0|         0|         0|           0|   pc_x_xy[2]
         0|         0|         0|           0|   pc_y_xx[2]
-0.00140548|-0.00140548|-0.00140548|           0|   pc_y_x1[2]
         1|         1|         1|           0|   pc_y_yy[2]
  0.997663|  0.997663|  0.997663|           0|   pc_y_y1[2]
         0|         0|         0|           0|   pc_y_xy[2]
         0|         0|         0|           0|   pc_x_xx[3]
   1.00365|   1.00365|   1.00365|           0|   pc_x_x1[3]
         0|         0|         0|           0|   pc_x_yy[3]
-0.00759572|-0.00759572|-0.00759572|           0|   pc_x_y1[3]
         0|         0|         0|           0|   pc_x_xy[3]
         0|         0|         0|           0|   pc_y_xx[3]
-0.000474801|-0.000474801|-0.000474801|           0|   pc_y_x1[3]
         0|         0|         0|           0|   pc_y_yy[3]
    1.0001|    1.0001|    1.0001|           0|   pc_y_y1[3]
         0|         0|         0|           0|   pc_y_xy[3]
  -54.5088|  -54.5088|  -54.5088|           0|   det_array_x
  -3.20477|  -3.20477|  -3.20477|           0|   det_array_y
  -0.18171|  -0.18171|  -0.18171|           0|   offx
```

```
 0.390044|  0.390044|  0.390044|           0|   offy
        1|         1|         1|           0|   flipx
       -1|        -1|        -1|           0|   flipy
      999|       999|       999|           0|   slit[1]
 -5.14691|  -5.14691|  -5.14691|           0|   slit[2]
 -2.98312|  -2.98312|  -2.98312|           0|   slit[3]
-0.821664| -0.821664| -0.821664|           0|   slit[4]
  1.33272|   1.33272|   1.33272|           0|   slit[5]
  3.49563|   3.49563|   3.49563|           0|   slit[6]
      999|       999|       999|           0|   slit[7]
      999|       999|       999|           0|   slit[8]
      999|       999|       999|           0|   slit[9]
```

C Abbreviations and acronyms

ANSI	American National Standards Institute
ASCII	American Standard Code for Information Interchange
CalibDB	Calibration Database
CPL	Common Pipeline Library
CPM	CRIRES Physical Model
CRIRES	Cryogenic High-Resolution IR Echelle Spectrograph
DFO	Data Flow Operations department
DFS	Data Flow System department
DMD	Data Management and Operations Division
DRS	Data Reduction System
ESO	European Southern Observatory
ESOREX	ESO-Recipe Execution tool
FITS	Flexible Image Transport System
FOV	Field Of View
FPN	Fixed Patter Noise
GUI	Graphical User Interface
ISAAC	Infrared Spectrometer And Array Camera
OB	Observation Block
OCA	Organisation Classification Association
QC	Quality Control
RON	Read Out Noise
SOF	Set Of Frames
UT	Unit Telescope
VLT	Very Large Telescope